Optimal Pebble Motion on a Tree[1]

Vincenzo Auletta and Pino Persiano

Dipartimento di Informatica ed Applicazioni, Università di Salerno, 84081 Baronissi, Italy
E-mail: auletta@unisa.it, giuper@unisa.it

Received December 21, 1995

In this paper we consider the following pebble coordination problem. Consider a tree with n vertices and k pebbles located at distinct vertices of the tree. Each pebble can be moved from its current position to an adjacent unoccupied vertex. Among the k pebbles, one distinguished pebble has been assigned a destination. We give an $O(n^5)$ algorithm for the problem of designing the shortest sequence of moves that takes the distinguished pebble from its original position to its destination. Our algorithm improves the running time of the best previously presented algorithm that needed to solve $O(n^6)$ min-cost flow problems on graphs of size $O(n)$. Our algorithm does not resort to reduction to flow but is instead based on a novel dynamic programming approach.

1. INTRODUCTION

In this paper we consider the following general pebble coordination problem. Let G be a graph with n vertices and k pebbles located at distinct vertices of the graph. Each pebble can be moved from its current position to an adjacent unoccupied vertex. Among the k pebbles, l distinguished pebbles have been assigned a destination.

In this context two kinds of questions can be asked. The first is a feasibility question; i.e., is there a sequence of moves at the end of which all distinguished pebbles end up at their designated destinations? Alternatively, one might ask for the shortest sequence of moves that results in all the distinguished pebbles being placed at their destinations. We denote by FEASIBLE(n, k, l) the computational problem that answers the first question and by OPTIMAL(n, k, l) the computational problem corresponding to the second question. For example, the problem FEASIBLE$(n, n-1, n-1)$ on a grid is a generalization of the well-known "15 puzzle."

Kornhauser *et al.* [3] exhibited a polynomial-time algorithm for the FEASIBLE(n, k, l) problem improving on earlier work by Wilson [7]. Recently, Auletta *et al.* [1] have given a linear time algorithm for FEASIBLE(n, k, K) for the special case of a tree. Papadimitriou *et al.* [4] gave a simple criterion to solve FEASIBLE when $l = 1$.

The computational problem OPTIMAL(n, k, l) appears to be substantially more difficult. Goldreich [2] proved that OPTIMAL$(n, n-1, n-1)$ is NP-complete for general graphs and, later, Ratner and Warmuth [5] proved that the problem remains NP-complete even when restricted to grids. The problem on general graphs remains NP-complete even if we only have one distinguished pebble (i.e., $l = 1$) [4]. On the other hand, in [4] a polynomial time algorithm has been given for the case in which the graph is a tree when there is only one distinguished pebble. The algoritm in [4] is based on a reduction of OPTIMAL$(n, k, 1)$ to a series of $O(n^6)$ min-cost flow problems on graphs of size $O(n)$.

Our result. In this paper we improve on the result of [4] for OPTIMAL$(n, k, 1)$ on trees and give a more efficient algorithm *Plan* to compute optimal plans. Our algorithm uses the properties of the canonical plans that are particular plans introduced in [4], but follows a quite different approach based on dynamic programming. The following theorem presents the main result of the paper.

THEOREM 1. *There exists an algorithm Plan that, for each instance of* OPTIMAL$(n, k, 1)$ *on a tree that admits a solution of finite cost, computes an optimal plan in time* $O(n + d^4 \cdot \min\{d^2, n\})$, *where n is the*

[1] A preliminary version of this paper appeared as A New Approach to Optimal Motion Planning on Trees with Obstacles *in* "Proceedings of 4-European Symposium on Algorithms," Lecture Notes in Computer Science, Vol. 1136, pp. 529–545. This work was partially supported by Italian Ministry of Scientific Research Project 40% "Algoritmi, Modelli di Calcolo e Strutture Informative."

0890-5401/00 $35.00

number of vertices of the tree and d is the distance from the initial position of the distinguished pebble to its destination.

Notice that $d = O(n)$ and thus the time complexity of *Plan* is $O(n^5)$. However, we stress that our algorithm runs in linear time for shallow trees (for example, on complete trees) and in general on instances in which the initial position of the distinguished pebble and its destination are not too far away. The same algorithm works also for edge-weighted trees and trees where vertices have capacities. In this case not all the edges have the same length and each vertex can host more than one pebble. In this case we try to minimize the total distance travelled by the robot and the obstacles.

The work of [4] was motivated mainly by problems in robot motion planning and referred to OPTIMAL$(n, k, 1)$ as the graph motion planning problem of 1 robot (GMP1R, in short). In fact, it is possible to recast OPTIMAL$(n, k, 1)$ as the problem of planning the shortest sequence of moves that takes a robot (the distinguished pebble) from its original position to its destination in the presence of movable obstacles (the remaining $k - 1$ pebbles). In this paper we will keep the robot-oriented terminology of [4]. Thus, we will refer to OPTIMAL$(n, k, 1)$ on a tree as a tree motion planning problem of 1 robot (TMP1R, in short), to the distinguished pebble as the robot, and to the remaining $k - 1$ pebbles as obstacles.

Organization of the paper. In Section 2 we set up our notations and review the concept of canonical plan introduced by [4]. In Section 3 we define the notion of *leftmost canonical plan* (LCP) and prove that each instance of OPTIMAL$(n, k, 1)$ that admits a finite cost solution has an optimal plan that is leftmost canonical. In Section 4 we present the ideas at the base of our approach to solving OPTIMAL$(n, k, 1)$. For ease of presentation, we do this by studying a simplified version of OPTIMAL$(n, k, 1)$ that we call STMP1R. We prove that solving an instance of STMP1R on a tree with n vertices can be reduced to solving a shortest path problem in a weighted DAG of size $O(n^7)$. An $O(n^7)$ algorithm for STMP1R follows from the well-known fact that the shortest path problem in a DAG can be solved in time linear to the size of the DAG. In Section 5, building on the ideas presented in the previous sections, we give a more efficient reduction that gives rise to a DAG of size $O(n^5)$. Finally, in Section 6 we give the technical details needed to extend the $O(n^5)$ algorithm for STMP1R to OPTIMAL$(n, k, 1)$.

2. NOTATION AND DEFINITIONS

A configuration *Conf* of the graph motion planning of 1 robot on a tree T (in the following referred to as TMP1R) consists of a subset O of the vertices of T containing obstacles and a vertex $s \notin O$ containing the robot. An instance $\mathcal{S}$ of TMP1R on a tree T consists of a configuration $Conf = (O, s)$, called the starting configuration, and a target vertex t. In Fig. 1, we give a pictorial representation of a configuration of the TMP1R; vertices with an obstacle are depicted in black, vertex s is the starting position of the robot, and vertex t is the target vertex. In the rest of the paper we say that a vertex is *full* if in the current configuration of the tree it contains an obstacle and *empty* otherwise. Empty vertices of *Conf* are called *holes*.

In [4] the following alternative definition of TMP1R is given. Suppose that a path P of the tree T from u to v is filled with obstacles except for v and suppose we move each of the obstacles in P one step towards v. Since the obstacles are indistinguishable, the net effect of these moves is equivalent

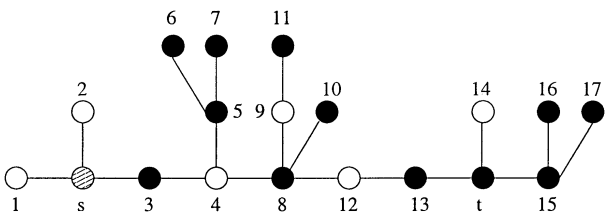

FIG. 1. An example of a instance of the TMP1R.

$\Pi = \{t \xrightarrow{s} 14, 8 \xrightarrow{s} 12, s \xrightarrow{R} 1,$
$3 \xrightarrow{1} 2, 1 \xrightarrow{R} 9, 12 \xrightarrow{9} 3,$
$13 \xrightarrow{9} 4, 9 \xrightarrow{R} t\}$

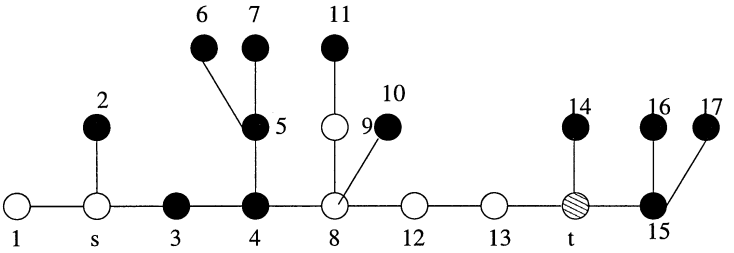

FIG. 2. A plan II for $\mathcal{S}$ and the configuration reached by applying plan II to $\mathcal{S}$.

to moving the obstacle initially at u to v. Note that this holds for the more general case where the path P may contain empty vertices other than v but the robot is at a vertex not in P. Thus, in the new formulation, an obstacle is allowed to traverse a path in one move provided that the path does not contain the robot and its destination is empty. In the remainder of the paper we call this move an obstacle move. The robot, instead, is allowed to traverse a path in one move only if all the vertices of the path are empty. In both the cases the cost of the move is given by the length of the traversed path.

More formally, we denote by $u \xrightarrow{v} w$ an obstacle move from vertex u to vertex w performed while the robot is at v and denote by $u \xrightarrow{R} w$ a robot move from u to w. We say that the obstacle move $u \xrightarrow{v} w$ is *legal* if w is an empty vertex and the path from u to w does not contain the robot; a robot move $u \xrightarrow{R} w$, instead, is legal if all the vertices of the path from μ to w are empty.

A plan Π for an instance $\mathcal{S}$ of TMP1R is a sequence of robot and obstacle moves. Plan Π is legal if all of its moves are legal. The *cost* of the plan is equal to the sum of the costs of its moves. The application of a plan to a configuration *Conf* gives a new configuration defined in the obvious way. We say that plan Π is *complete* for an istance $\mathcal{S} = (Conf, t)$ if applying the plan Π to the starting configuration *Conf* yields a new configuration with the robot at the target vertex t. In Fig. 2 a plan Π for the instance of Fig. 1 is given, and it is depicted the final configuration reached by Π. The cost of Π is 20 and, since the robot ends up at vertex t, the plan is also complete.

The TMP1R problem on instance $\mathcal{S}$ consists of computing a complete plan of minimum cost for $\mathcal{S}$.

Throughout this paper n denotes the number of vertices of T and d denotes the length of the path $P = (v_0, \ldots, v_d)$ from the source vertex $s = v_0$ to the target vertex $t = v_d$. We call P the *critical path* of $\mathcal{S}$ and call each vertex of P, distinct from s, that has at least one neighbor not on P a branch vertex. For each vertex v_j of the critical path we denote by B_j the set of vertices of T distinct from v_j that are connected to v_j through a path not containing vertices of P. For each non empty B_j call one of the vertices of B_j that is adjacent to v_j (we select one of the neighbors of v_j that is closest to a hole of B_j) a *sidestep vertex* of v_j. It can be easily seen that $P, B_0, B_1, \ldots, B_d$ define a partition of the set of vertices of T. In Fig. 3 it is shown how the tree of Fig. 1 is partitioned in P, B_0, B_2, B_3, and B_6.

Moreover, for each vertex v_j in P and for each branch vertex v_r, with $0 < r < j$, we set $\mathrm{Ch}(r, j) = \{v_h \mid r \leq h < j\}$, $\mathrm{Ch}(0, j)$, instead, denotes the set $B_0 \cup \{v_h \mid 0 \leq h < j\}$.

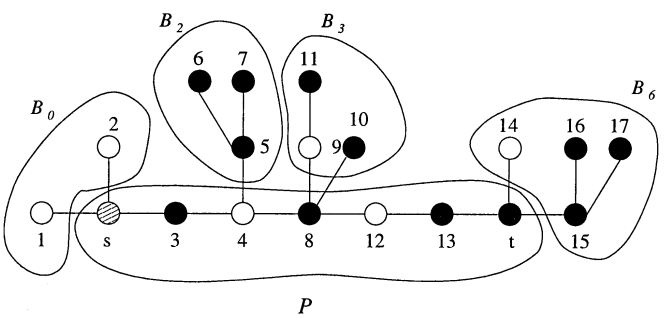

FIG. 3. The partition of the tree T of Fig. 1. $P = (s, 3, 4, 8, 12, 13, t)$; vertices 4, 8 and t are branch vertices; $B_0 = \{1, 2\}$, $B_2 = \{5, 6, 7\}$, $B_3 = \{9, 10, 11\}$, $B_6 = \{14, 15, 16, 17\}$.

Finally, we define the following partial order relation on the vertices of T with respect to the instance $\mathcal{S}$. For each pair of vertices u, w we say that

$$u <_{\mathcal{S}} w \text{ if and only if } \begin{cases} u = v_i, w = v_j & \text{with } i < j \\ u = v_i, w \in B_j & \text{with } i < j \\ u \in B_i, w = v_j & \text{with } i \leq j \\ u \in B_i, w \in B_j & \text{with } i < j \\ u, w \in B_0 \text{ and } w \text{ is on the path from } u \text{ and } s. \end{cases}$$

Notice that if u and w are both in B_i, for $i > 0$, then they are unrelated. In the following we omit the name of the instance when it is clear from the context. Moreover, with a little abuse of denotation we say that $u \leq w$ meaning that either $u = w$ or $u < w$.

2.1. *Canonical Plans*

Canonical plans are a particular type of plans described in [4]. To introduce the definition of canonical plans we need a few definitions given in [4].

We call a sequence of moves that takes the robot from the source vertex to the target vertex *quasi-monotone* if the robot, on arriving at an internal vertex v of P, either moves directly to the next vertex in P or, if v is a branch vertex, moves first to the sidestep vertex of v and then comes back to v and proceeds to the next vertex of P.

A *quasi-bitonic* plan consists of two parts: a *backup part*, in which the robot visits only vertices of B_0 and arrives at s; and a *forward part*, that is a quasi-monotone plan that takes the robot from s to t.

Let P_{for} and P_{bac} be the set of vertices visited by the robot during the forward and backup part of a quasi-bitonic plan, respectively. Suppose that an obstacle is moved during the plan from vertex $v \in \mathrm{P}_{for}$ to w, while the robot is at z. The move $v \xrightarrow{z} w$ is classified as:

- *outward* if $w \notin B_0 \cup \mathrm{P}_{for}$;
- *forward* if $w \in \mathrm{P}_{for}$ and $v < w$;
- *backward* if $w \in B_0$ or if $w \in \mathrm{P}_{for}$ and $w < v$.

DEFINITION 1 [4]. A canonical plan is a quasi-bitonic plan that satisfies the following additional conditions on the obstacle moves:

1. no obstacle placed outside P_{for} and P_{bac} is ever moved;
2. each obstacle initially at a vertex of P_{for} is moved once (outward or backward) or twice (first forward and then backward);
3. if an obstacle is moved backward, then it traverses at least a branch vertex;
4. each obstacle initially at a vertex of P_{bac} is moved out of P_{bac} before the robot leaves s;
5. all forward and outward moves are performed before the robot leaves s.

The following lemma holds for canonical plans.

LEMMA 1 [4]. *For each instance of TMP1R that admits a solution of finite cost, there exists an optimal plan that is canonical.*

In this paper we consider only canonical plans. It is easy to see that a canonical plan performs all forward and outward moves while the robot is at s and backward moves while the robot is at the sidestep of a branch vertex. Therefore, in the following we omit the position of the robot in denoting forward and outward moves and denote by "$v \xrightarrow{i} w$" the backward move from vertex v to vertex w that takes place while the robot is at the sidestep of v_i.

3. LEFTMOST CANONICAL PLANS

In this section we introduce the notion of Leftmost Canonical Plan (LCP), which is a particular type of canonical plan, and prove that for each instance $\mathcal{S}$ of the TMP1R that admits a finite cost solution there exists an optimal plan that is leftmost canonical. Therefore, in the rest of the paper we will restrict our attention to LCP.

DEFINITION 2. A canonical plan A for an instance $\mathcal{S} = ((O, s), t)$ of TMP1R is a leftmost canonical plan if and only if the following conditions hold.

1. Plan A performs all forward moves before any outward moves.

2. If A contains a backward move $w \xrightarrow{i} w'$ then i is the smallest integer such that $w' < v_i$ and A contains a robot move ending at the sidestep of v_i.

3. If A contains a forward move $u \to u'$ and a backward move $u' \xrightarrow{i} u''$ then $u \leq v_i < u'$.

4. If A contains a move $u \to u'$, with $u < u'$, then A does not contain any outward move $w \to w'$ with $u < w$ and $w' < u'$.

5. If A contains a backward move $w \xrightarrow{i} w'$ and a forward move $u \to u'$, with $u \leq v_i < w < u'$, then, for each vertex $z \notin O$ and $v_i < z < w$, A contains a forward move ending at z.

6. If A contains a backward $w \xrightarrow{i} w'$ then for each $v_i < u < w$ A contains a move originating at u and ending at a vertex $u' < v_i$.

THEOREM 2. *For each instance $\mathcal{S}$ of TMP1R that admits a finite cost solution there exists an optimal plan that is leftmost canonical.*

Proof. Our proof proceeds in six steps. We start with an optimal canonical plan A for $\mathcal{S}$ and in step i we modify the plan to enforce property i of Definition 2. That is, in step i we prove that there exists a canonical plan A_i of cost not greater than A for which properties $1, \ldots, i$ hold. In this way, we obtain an optimal plan A_6 for $\mathcal{S}$ which is leftmost canonical and thus we prove the theorem. ■

CLAIM 1. *There exists an optimal canonical plan for $\mathcal{S}$ that enjoys property* 1 *of LCP.*

Proof. Let A be an optimal canonical plan for $\mathcal{S}$ (such a plan exists by Lemma 1) and consider the plan A' obtained from A through the following two-step transformation.

Step 1. Replace all pairs of moves consisting of a forward move $u \to u'$ and an outward move $u' \to u''$ with the outward move $u \to u''$;

Step 2. Permute the execution order of the outward and forward moves obtained after Step 1 by performing first all the forward moves and then the outward moves.

We remark that the first step of the transformation is necessary to guarantee that plan A' is legal: in fact, had we just inverted the order of the forward and outward moves, we might have had a forward move ending at a nonempty vertex (see Fig. 4).

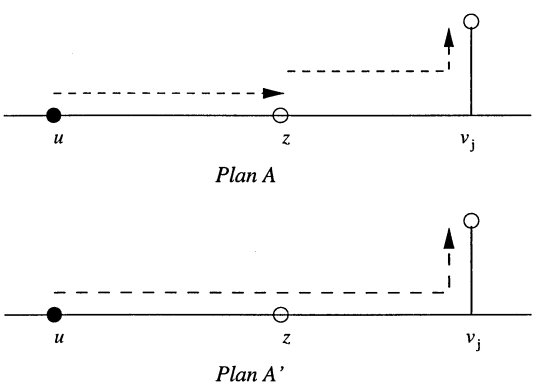

FIG. 4. Transformation of a canonical plan into a canonical plan of not greater cost that enjoys property 1 of Definition 2.

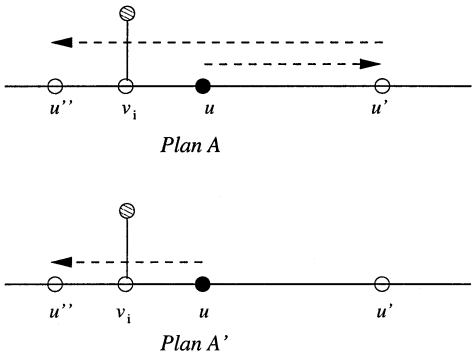

FIG. 5. The transformation of a canonical plan that enjoys properties 1 and 2 of LCP to a not greater cost plan that enjoys properties 1 to 3.

A' is a legal canonical plan since all the new moves added to A are outward moves traversing vertices not containing the robot and ending at empty vertices. The cost of A' is clearly no greater than the cost of A and A' enjoys property 1. ■

CLAIM 2. *There exists an optimal canonical plan for $\mathcal{S}$ that enjoys properties* 1 *and* 2 *of LCP.*

Proof. Let A be an optimal canonical plan for $\mathcal{S}$ that enjoys property 1 of Definition 2 (by Claim 1 such a plan exists). For each vertex $u \in \mathrm{P}_{for}$ denote by $b(u)$ the smallest integer i such that $u < v_i$ and A contains a robot move ending at the sidestep of v_i.

Consider the canonical plan A' obtained from A by replacing each backward move $w \xrightarrow{i} w'$ of A with the backward move $w \xrightarrow{b(w')} w'$. It can be easily seen that plan A' is a legal canonical plan of cost no greater than A and that it enjoys properties 1 and 2 of LCP. In fact we substitute each backward move of A with another backward move that has the same origin and destination (and thus the same cost) and differs only on the sidestep vertex where the robot is placed when the move occurs. ■

CLAIM 3. *There exists an optimal canonical plan for $\mathcal{S}$ that enjoys properties* 1–3 *of LCP.*

Proof. Let A be an optimal canonical plan for $\mathcal{S}$ that enjoys properties 1 and 2 of LCP (by Claim 2, such a plan exists). We construct a new plan A' from A by replacing each pair of moves of A consisting of a forward move $u \to u'$ and a backward move $u' \xrightarrow{i} u''$, where $v_i < u$, with the backward move $u \xrightarrow{i} u''$ (see Fig. 5).

We observe that the new backward move is legal, as it ends at a vertex that is the destination of a move in A (and thus it is empty), and it is performed while the robot is at a sidestep vertex. Moreover, it traverses branch vertex v_i. Thus, A' is a legal canonical plan and its cost is not greater than A.

To prove that A' enjoys properties 1–3 of LCP we observe that the plan is obtained from A, which enjoys the first two properties, by eliminating some forward moves and changing the origin of some backward moves. Thus, A' enjoys properties 1 and 2, since the order in which moves are performed is the same as A. It also enjoys property 3 since for each pair of moves of $A' u \to u'$ and $u' \xrightarrow{i} u''$ we have that $u \le v_i < u'$. In fact, were $v_i < u$ then these moves would have been replaced by the transformation of A into A'; on the other hand, were $u' \le v_i$, then plan A would not be legal since the robot would have visited u' while the vertex is occupied by an obstacle. ■

CLAIM 4. *There exists an optimal canonical plan for $\mathcal{S}$ that enjoys properties* 1–4 *of LCP.*

Proof. We say that a pair of vertices (u, w) is 4-*problematic for plan* A if $u < w$ and A contains an obstacle move from u to u', with $u < u'$, and an outward move $w \to w'$, with $w' < u'$. It can be easily seen that a plan that has no 4-problematic enjoys property 4 of Definition 2.

We next show how to transform an optimal canonical plan A that enjoys properties 1 to 3 of LCP (by Claim 3 such a plan exists) and has $h > 0$ 4-problematic pairs into an equivalent canonical plan A' of no greater cost that enjoys properties 1 to 3 and has less than h 4-problematic pairs. Applying iteratively the same transformation to plan A' we obtain a new optimal canonical plan $\mathcal{S}$ that enjoys properties 1 to 4 of LCP.

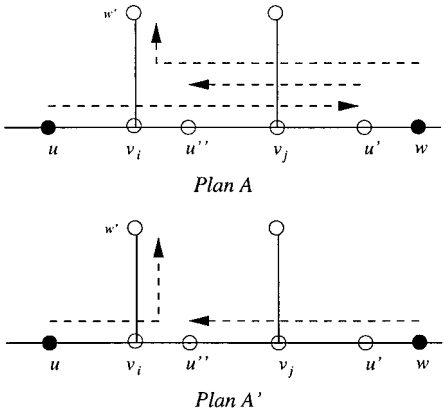

FIG. 6. The transformation of a canonical plan A that has a 4-problematic pair (u, w) into a not greater cost canonical plan A' for which (u, w) is not 4-problematic. In this case u is moved forward to a vertex $u' < w$.

Let u, w be vertices of P_{for} such that (u, w) is a 4-problematic pair for A; for each 4-problematic pair (x, y) we have that $y \leq w$; for each 4-problematic pair (x, w) we have that $u \leq x$.

In order to obtain A' from A we distinguish three cases, depending on the type of move originating at u and on the positions of u' and w.

Case I: $u \to u'$ is a forward move and $u' < w$.

Since $u \to u'$ is a forward move then A also contains a backward move $u' \xrightarrow{i} u''$, originating at u'. We obtain A' by replacing the moves $u \to u'$, $u' \xrightarrow{i} u''$, and $w \to w'$ with the outward move $u \to w'$ and the backward move $w \xrightarrow{i} u''$ (see Fig. 6).

Plan A' is easily seen to be legal, canonical, and of cost not greater than A. Properties 1 to 3 of LCP are maintained since we have only replaced an outward move with another outward move ending at the same vertex and changed the origin of a backward move to a vertex belonging to O.

Case II: $u \to u'$ is a forward move and $w < u'$.

As in the previous case, A contains a backward move $u' \xrightarrow{i} u''$, originating at u'. We distinguish two subcases, depending on the position of v_i.

II.B: $w \leq v_i$.

In this case $u \to u'$ and $w \to w'$ are replaced by the outward move $u \to w'$ and the forward move $w \to u'$ (see Fig. 7b).

As before, it is easy to see that A' is a legal canonical plan and its cost is not greater than A. Properties 1 and 2 of LCP are obviously maintained. For Property 3 we observe that a new forward ending at u' is

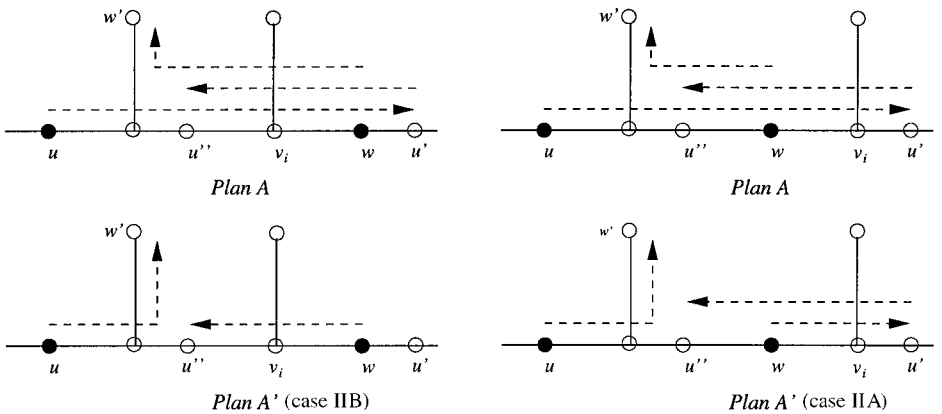

FIG. 7. The transformation of a canonical plan A that has a 4-problematic pair (u, w) into a not greater cost canonical plan A' for which (u, w) is not 4-problematic. In this case u is moved forward to a vertex u' such that $w < u'$.

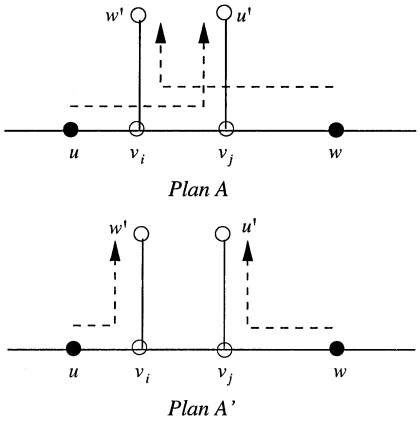

FIG. 8. The transformation of a canonical plan A that has a 4-problematic pair (u, w) into a not greater cost plan A' for which (u, w) is not 4-problematic. In this case u is moved outward.

added to the plan, but the corresponding backward move is $u' \xrightarrow{i} u''$, with $w \leq v_i < u'$. Thus A' also satisfies Property 3.

II.B: $v_i < w$.

In this case $u \to u'$, $u' \xrightarrow{i} u''$, and $w \to w'$ are replaced by the outward move $u \to w'$ and the backward move $w \xrightarrow{i} u''$ (see Fig. 7a).

The proof is similar to the previous case. We have only to observe that in this case we have added a backward move. But, since the origin of this move belongs to O, Property 3 holds.

Case III: $u \to u'$ is an outward move.

We obtain A' by replacing the moves $u \to u'$ and $w \to w'$ with the outward moves $u \to w'$ and $w \to u'$ (see Fig. 8).

The proof is similar to the previous cases.

It remains to prove that the number of 4-problematic pairs for A' is strictly less than the number of 4-problematic pairs for A. We will prove that if a pair is 4-problematic for A' then it is 4-problematic for A too. Since the pair (u, w) is obviously not 4-problematic for A', the claim follows.

Consider a 4-problematic pair (x, y) for A' and distinguish the following cases.

—$x \neq u$ and $y \neq w$

(x, y) is clearly 4-problematic for A too.

—$x = u$ and $y \neq w$

If (u, y) is 4-problematic for A' then $u < y$ and A' contains the outward move $y \to y'$, with $y' < w'$.

Let us look at plan A. Plan A contains the move $u \to u'$, with $u < u'$, and the outward move $y \to y'$, with $w' < u'$. Then we have that $y' < w' < u'$ and (u, y) is 4-problematic for A too.

—$y = w$ and $x \neq u$

If (x, w) is 4-problematic for A' then $x < w$ and w is the origin of an outward move. Therefore, in obtaining A' from A we have used transformation of case III and A' contains a move $x \to x'$, with $x < x'$ and $u' < x'$.

Plan A, instead, contains the moves $x \to x'$ and $w \to w'$. Since $w' < u' < x'$ we have that (x, w) is 4-problematic for A too.

—$x = w$ and $y \neq u$

Suppose (w, y) is 4-problematic for A'. Then $w < y$ and w cannot be the origin of a backward move. In this case A' has been obtained from A using the transformation of Cases II.A or III. In both cases A' contains a move $y \to y'$, with $y' < u'$.

Then plan A contains moves $u \to u'$ and $y \to y'$. Since $u < w < y$ we have that (u, y) is 4-problematic for A. But, this contradicts the hypothesis that there is no vertex y such that $w < y$ and is part of a 4-problematic pair for A. Thus, there is no 4-problematic pair (w, y) for A'.

—$y = u$ and $x \neq w$

Suppose (x, u) is 4-problematic for A'. Then $x < u$ and A' contains the move $x \to x'$, with $x < u$ and $w' < x'$.

Plan A, instead, contains the moves $x \to x'$ and $w \to w'$. Since $x < u < w$ then (x, w) is 4-problematic for A. But, this contradicts the hypothesis that there is no vertex $x < u$ such that (x, w) is a 4-problematic pair with w. Thus, there is no 4-problematic pair (x, u) for A'. ■

CLAIM 5. *There exists an optimal canonical plan for $\mathcal{S}$ that enjoys properties* 1–5 *of LCP.*

Proof. Let (u, z, w) be a triplet of vertices of P_{for} such that $u < z < w$. We say that the triplet (u, z, w) is 5-*problematic for plan A* if A contains a forward move $u \to u'$, with $w < u'$, a backward move $w \xrightarrow{i} w'$, with $u \le v_i < z$, and no move originating at z. It can be easily seen that if a plan has no 5-problematic triplet then it enjoys property 5 of Definition 2.

As in Claim 4 we next show how to transform any optimal canonical plan A for $\mathcal{S}$ that enjoys properties 1 to 4 of LCP (by Claim 4 such a plan exists) and has $h > 0$ 5-problematic triplets into an optimal canonical plan A' for $\mathcal{S}$ that enjoys properties 1 to 4 of LCP and has less than h 5-problematic triplets. Applying iteratively the same transformation to plan A' we obtain an optimal canonical plan $\mathcal{S}$ that enjoys properties 1 to 5 of LCP.

Let u, z, and w be vertices of P_{for} such that (u, z, w) is a 5-problematic triplet for A; for each 5-problematic triplet (x, y, t) we have that $t \le w$; for each 5-problematic triplet (x, y, w) we have that $z \le y$. In describing the transformation we distinguish two cases depending on the position of w.

Case I: $v_j < w$.

Since $u \to u'$ is a forward move then plan A contains a backward move $u' \xrightarrow{j} u''$, originating at u'.

Then we replace moves $u \to u'$, $u' \xrightarrow{j} u''$, and $w \xrightarrow{i} w'$ with $u \to z$, $w \xrightarrow{j} u''$, and $z \xrightarrow{i} w'$ (see Fig. 9).

Case II: $w \le v_j$.

We distinguish two subcases depending on whether $w \in O$:

Case A: $w \in O$

We replace moves $u \to u'$, $u' \xrightarrow{j} u''$, and $w \xrightarrow{i} w'$ with $u \to z$, $w \to u'$, $u' \xrightarrow{j} u''$, and $z \xrightarrow{i} w'$ (see Fig. 10a).

Case B: $w \notin O$

The plan A contains a forward move $w'' \to w$ ending at w. Then, we replace moves $u \to u'$, $u' \xrightarrow{j} u''$, $w'' \to w$ and $w \xrightarrow{i} w'$ with $u \to z$, $z \xrightarrow{i} w'$ $w'' \to u'$ and $u' \xrightarrow{j} u''$ (see Fig. 10b).

From Figs. 9 and 10 it can be easily seen that A' is a legal canonical plan of cost not greater than A that enjoy properties 1–4 of LCP. In fact, properties 1 and 2 clearly hold since we have obtained A' from A by replacing some forward moves and by changing the origin of some backward moves (and thus the order of the moves is not changed with respect to A); the forward and backward moves added

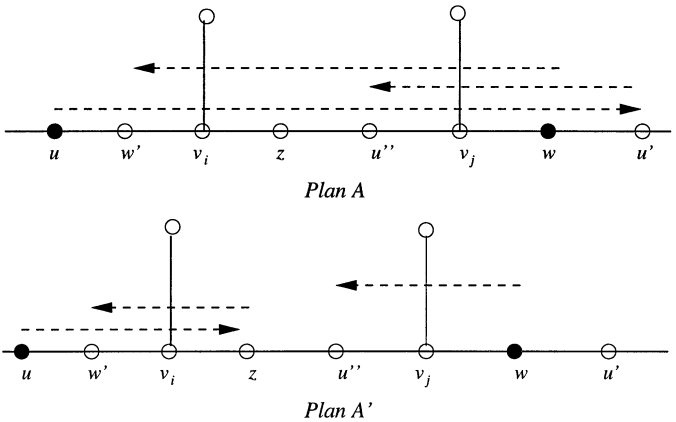

FIG. 9. The transformation of a canonical plan A that has a 5-problematic triplet (u, z, w) to a not greater cost canonical plan A' for which (u, z, w) is not 5-problematic. Case $v_j < w$.

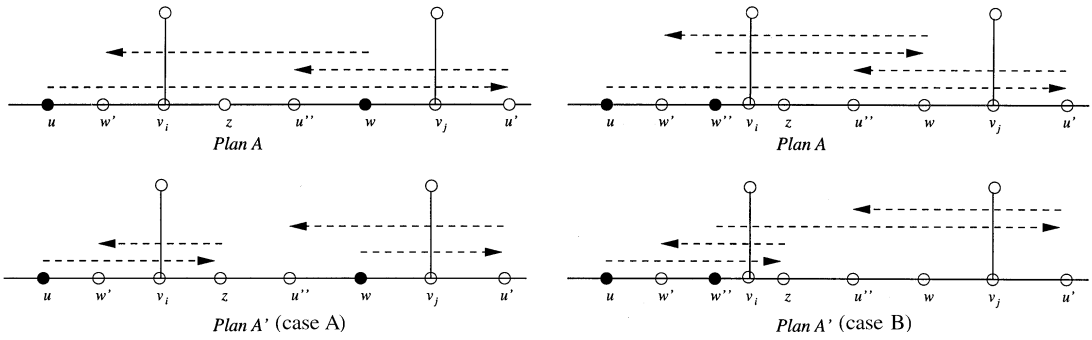

FIG. 10. The transformation of a canonical plan A that has a 5-problematic triplet (u, z, w) to a not greater cost canonical plan A' for which (u, z, w) is not 5-problematic. Case $w \leq v_i$.

to A' enjoy property 3; there is no 4-problematic pair for A' since by the hypothesis in A there is no outward move originating at a vertex z, with $u < z$, and ending at a vertex $z' < u'$.

It remains to prove that A' has less than h 5-problematic triplets. As for the previous claim, we will prove this by showing that any triplet that is 5-problematic for A' is 5-problematic for A, too. Since (u, z, w) is 5-problematic for A but not for A', the claim follows.

We start by observing that A' has no 5-problematic triplet of the form $(\overline{u}, z, \overline{w})$ (in A', z is the destination of a forward move) or of the form $(\overline{u}, \overline{z}, w)$ (in A', w is not the origin of a backward move).

Let (x, y, t) be a 5-problematic triplet for A'. By the definition of 5-problematic triplet we have that x is the origin of a forward move (and thus $x \neq z$), t is the origin of a backward move (and thus $t \neq u, w$), and y is not the origin of any move (and thus $y \neq u, z, w$). Consider all the possible cases:

—(x, y, t) does not contain any of u, z, and w

Clearly, (x, y, t) is 5-problematic for A, too.

—$t = z$

Plan A' contains the backward move $z \xrightarrow{i} w'$.

If (x, y, z) is 5-problematic for A' then A' does not contain any move originating at y. It follows that A does not contain any move originating at y, contradicting the hypothesis that all the vertices of P_{for} between v_i and z are origin of obstacle moves in A.

—$x = w$

Plan A' contains the outward move $w \rightarrow u'$ and the backward move $t \xrightarrow{j} t'$.

If (w, y, t) is 5-problematic for A' then it is $w < v_j < t < u'$. Then (u, y, t) is 5-problematic for A, contradicting the hypothesis that there is no vertex $t > w$ that is part of a 5-problematic triplet for A.

—$x = u$

Plan A' contains the forward move $u \rightarrow z$ and the backward move $t \xrightarrow{j} t'$.

If (u, y, t) is 5-problematic for A' then it is $t < z$. Since by hypothesis $z < u'$ and plan A contains the moves $u \rightarrow u'$ and $t \xrightarrow{j} t'$, we have that (u, y, t) 5-problematic for A, too. ■

CLAIM 6. *There exists an optimal canonical plan for $\mathcal{S}$ that enjoys properties* 1–6 *of LCP.*

Proof. We say that a pair of vertices (u, w) is 6-*problematic for a plan A* if $u < w$ and A contains a backward move $w \xrightarrow{i} w'$ and a move $u \rightarrow u'$, where $v_i < u$ and $v_i \leq u'$. It can be easily seen that if a plan has no 6-problematic pairs for property 6 then it enjoys property 6 of LCP.

As in previous claims we next show how to transform any optimal canonical plan A for $\mathcal{S}$ that enjoys properties 1 to 5 of LCP (by Claim 5 such a plan exists) and has $h > 0$ 6-problematic pairs into an optimal canonical plan A' for $\mathcal{S}$ that enjoys properties 1 to 6 of LCP and has less than h 6-problematic pairs. Applying iteratively the same transformation to plan A' we obtain an optimal canonical plan $\mathcal{S}$ that enjoys properties 1 to 6 of LCP.

Let u, w be vertices of P_{for} such that (u, w) is a 6-problematic pair for A; for each 6-problematic pair (x, y) we have that $t \leq w$; for each 6-problematic pair (x, w) we have that $u \leq x$. We obtain plan A' by replacing moves of A originating at u and w. We distinguish four cases depending on the kind of move $u \rightarrow u'$ of A.

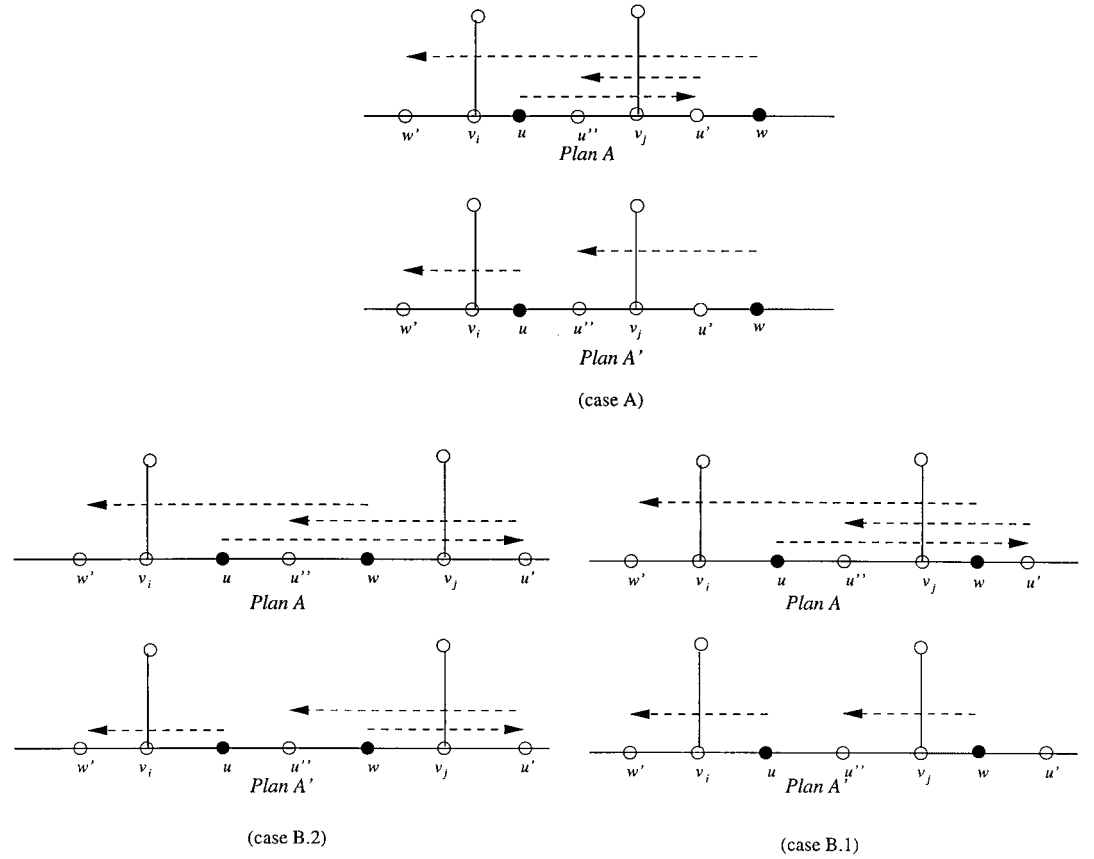

FIG. 11. The transformation of a canonical plan A that has a 6-problematic pair (u, w) to a not greater cost plan A' for which (w, w) is not 6-problematic. We consider the case in which an obstacle at u is moved forward and distinguish two cases depending on the position of u' and w: Case A shows the transformation applied when the obstacle at u is moved forward to a vertex $u' < w$. Case B.1 shows the transfomation applied when the obstacle at u is first moved forward to a vertex $u' \geq w$ and then backward, while the robot is at the sidestep of a branch vertex $v_j < w$; case B.2 shows the transfomation applied when the obstacle at u is first moved forward to a vertex $u' \geq w$ and then backward, while the robot is at the sidestep of a branch vertex $v_j \geq w$.

Case A: The obstacle at u is moved forward to a vertex u' such that $u < u' < w$.

Since A is canonical it contains a backward move $u' \xrightarrow{j} u''$. We replace the moves $u \rightarrow u'$, $u' \xrightarrow{j} u''$, and $w \xrightarrow{i} w'$ with the backward moves $u \xrightarrow{i} w'$ and $w \xrightarrow{j} u''$ that have no greater cost (see Fig. 11 case A).

From Figure 11 it can be easily seen that the new plan is a legal canonical plan and its cost is not greater than A. Moreover, it enjoys properties 1 to 5 of LCP. In fact, we have obtained A' by eliminating a forward move and by changing the origins of two backward moves. Thus, properties 1, 2, 4, and 5 trivially hold. To show that property 3 holds we observe that the only new move that can violate the property is $w \xrightarrow{j} u''$ (in fact $u \in O$ and the move $u \xrightarrow{i} w'$ enjoys the property). But, since by hypothesis A enjoys the property, it must be the case that w is the destination of a forward move $w'' \rightarrow w$, with $w'' \leq v_i < w$. In A' the obstacle at w is moved when the robot is at the sidestep of $v_j > v_i$ and thus the backward move originating at w does not violate property 3.

It remains to prove that the number of 6-problematic pairs for A' is strictly less than h. We show that each pair of vertices (x, y) that is 6-problematic for A' is 6-problematic for A too. Since (u, w) is obviously not 6-problematic for A' the claim follows. We have to distinguish the following cases.

—$x \neq u$ and $y \neq w$

(x, y) is clearly 6-problematic for A too.

—$x = u$ and $y \neq w$

If (u, y) is 6-problematic for A' then $u < y$ and A' contains the backward moves $y \xrightarrow{k} y'$ and $u \xrightarrow{i} w'$, with $v_k < u < y$ and $v_k \leq w'$.

Plan A, instead, contains the moves $y \xrightarrow{k} y'$ and the forward move $u \to u'$.

Then we have that $v_k < u < u'$ and (u, y) is 6-problematic for A too.

—$x \neq u$ and $y = w$

If (x, w) is 6-problematic for A' then $x < w$ and A' contains the backward move $w \xrightarrow{j} u''$ and the move $x \to x'$, with $v_j < x < w$ and $v_j \leq x'$.

Plan A, instead, contains the backward moves $w \xrightarrow{i} w'$, $u' \xrightarrow{j} u''$ and the forward move $u \to u'$. We observe that, since by hypothesis A enjoys property 3 of LCP, then $u \leq v_j$ and thus $v_i < u \leq v_j$. Then, we obtain that $v_i < v_j < x$ and $v_i < v_j \leq x'$ and thus (x, w) is 6-problematic for A too.

—$x = w$ and $y \neq u$

If (w, y) is 6-problematic for A' then $w < y$ and A' contains the backward moves $y \xrightarrow{k} y'$ and $w \xrightarrow{j} u''$, with $v_k < w < y$ and $v_k \leq u''$.

Plan A, instead, contains the moves $u' \xrightarrow{j} u''$ and $y \xrightarrow{k} y'$. We observe that (u', y) is 6-problematic for A. In fact, since $u' < w < y$ and $v_k \leq u'' < u'$, we have that $v_k < u'$ ($v_k \leq u''$ by hypothesis). But this contradicts the hypothesis that there is no vertex $y > w$ that is part of a 6-problematic pair for A. Therefore, there is no pair (w, y) that is 6-problematic for A'.

—$x \neq w$ and $y = u$

If (x, u) is 6-problematic for A' then $x < u$ and A contains the backward move $u \xrightarrow{i} w'$ and the move $x \to x'$, with $v_i < x < u$ and $v_i \leq y'$.

Plan A, instead, contains the moves $w \xrightarrow{i} w'$ and $x \to x'$. Since $u < w$ we have that $v_i < x < w$ and $v_i \leq x'$. Thus, (x, w) is 6-problematic for A. But this contradicts the hypothesis that there is no vertex $x < u$ such that (x, w) is 6-problematic for A. Therefore, there is no pair (x, u) that is 6-problematic for A'.

Case B: The obstacle at u is first moved forward to a vertex $u' \geq w$ and then backward to u'', while the robot is at the sidestep vertex of v_j.

In this case the transformation depends on the position of v_j: if $w \leq v_j$ then we replace the moves $w \xrightarrow{i} w'$ and $u \to u'$ with the moves $w \to u'$, $u \xrightarrow{i} w'$ (see Fig. 11 case B.1); if $v_j < w$, instead, we replace the moves $w \xrightarrow{i} w'$, $u \to u'$, and $u' \xrightarrow{j} u''$ with the moves $u \xrightarrow{i} w'$ and $w \xrightarrow{j} u'$ (see Fig. 11 case B.2).

This case is treated similar to case A.

Case C: The obstacle at u is moved backward to u', while the robot is at the sidestep vertex of v_j.

In this case we replace the backward moves $w \xrightarrow{i} w'$ and $u \xrightarrow{j} u'$ with the backward moves $u \xrightarrow{i} w'$ and $w \xrightarrow{j} u'$ (see Fig. 12 case C).

Arguments similar (but simpler) than the ones used for case A show that the A' is an optimal canonical plan for $\mathcal{S}$ that enjoy properties 1 to 5 and it has less than h 6-problematic pairs.

Case D: The obstacle at u is moved outward to u'.

In this case we replace the moves $w \xrightarrow{i} w'$ and $u \to u'$ with the moves $u \xrightarrow{i} w'$ and $w \to u'$ (see Fig. 12 case D).

This case is treated similarly to case C.

4. A SIMPLER MOTION PLANNING PROBLEM

In this section we consider a simplified version of our problem, that we call STMP1R, and give a simple $O(n^7)$ algorithm for it. Although inefficient and for a simplified problem, this algorithm contains all the ideas at the base of the $O(n^5)$ algorithm for TMP1R. We postpone efficiency considerations to the next section where we show how to improve the algorithm for STMP1R to obtain a running time of $O(n^5)$. Finally, in Section 6 we discuss the technical modifications needed to obtain the $O(n^5)$ algorithm for TMP1R.

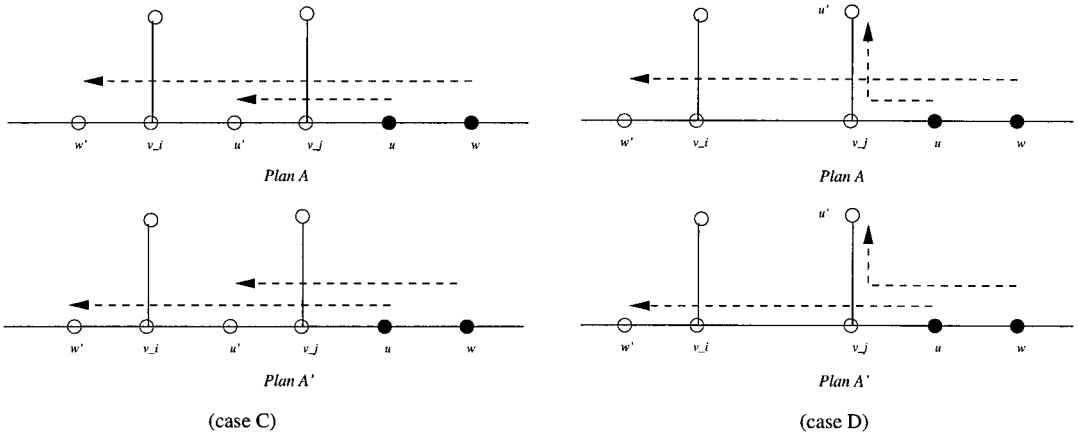

FIG. 12. The transformation of a canonical plan A that has a 6-problematic pair (u, w) to a not greater cost plan A' for which (u, w) is not 6-problematic. We consider the case in which obstacle at u is moved outward or backward.

STMP1R differs from TMP1R in two respects: first, the robot is not allowed to visit any vertex of B_0 distinct from the sidestep of s (and thus the backup part of the plan is empty); second, obstacle moves cannot start or end at sidestep vertices (and thus in the starting configuration all sidestep vertices are empty). As we shall see in Section 6, dealing with these two issues is only a question of technicalities. It is easily seen that, similar to TMP1R, any instance of STMP1R that admits a solution of finite cost has an optimal leftmost canonical plan.

At the base of our approach is the idea of splitting the set of moves of a complete plan A for the instance $\mathcal{S}$ of the STMP1R into two sets $L(A, j)$ and $R(A, j)$ with respect to a vertex v_j of the critical path,

DEFINITION 3. Let $\mathcal{S}$ be an instance of the STMP1R problem, let $P = (v_0, \ldots, v_d)$ be its critical path, and let A be a LCP that is complete for $\mathcal{S}$. For each $0 \le j \le d$, we partition A into a left part $L(A, j)$ and a right part $R(A, j)$ with respect to v_j according to the following rules. For each move of A from u to w we distinguish several cases depending on the type of the move:

Case 1: $u \xrightarrow{R} w$ is a robot move.
If $w \le v_j$ then $u \xrightarrow{R} w$ belongs to $L(A, j)$; else it belongs to $R(A, j)$.

Case 2: $u \to w$ is an outward move.
Let $w \in B_i$.
If $v_i \le v_j$ then $L(A, j)$ contains $u \to w$.
If $v_j < v_i$ and $v_j < u$ then $R(A, j)$ contains $u \to w$.
If $u < v_j < v_i$ then $L(A, j)$ contains $u \to v_j$ and $R(A, j)$ contains $v_j \to w$.

Case 3: $u \xrightarrow{i} w$ is a backward move.
Let v_r be defined as follows. If plan A has not taken any sidestep before arriving at v_j then $v_r = s$; else v_r is the last branch vertex at which the robot has taken a sidestep before arriving at v_j.
If $v < v_r$, then $L(A, j)$ contains $u \xrightarrow{i} w$ else $R(A, j)$ contains $u \xrightarrow{i} w$.

Case 4: $u \to w$ is a forward move.
Let v_r be defined as before and define l as the largest integer such that $j \le l$ and v_l is the origin of a move ending at a vertex z, where either $z < v_r$ or $z \in B_h$ with $h \le j$.
If $w \le v_l$ (and thus $u \le v_l$) then $L(A, j)$ contains $u \to w$.
If $v_l < u$ (and thus $v_l < v$) then $R(A, j)$ contains $u \to w$.
If $u < v_j$ and $v_l < w$ then $L(A, j)$ contains $u \to v_j$ and $R(A, j)$ contains $v_j \to w$.

We say that a set of moves R is a *legal completion* of a left part L with respect to S if there exists a vertex $v_j \in P$ and a LCP A that is complete for S such that $L = L(A, j)$ and $R = R(A, j)$. The

cost of the legal completion R is simply the sum of the costs of the moves in R. The minimum legal completion R^* of L is the legal completion of L with minimum cost.

Let $\mathcal{S}$ be an instance of STMP1R. For each vertex v_j, we partition the set

$$\{L(A, j) \mid A \text{ is a complete LCP for } \mathcal{S}\}$$

into classes according to five parameters. We say that the left part $L(A, j)$ of A relative to v_j belongs to the class $\mathcal{C}(j, k, l, f, r)$ if it contains k obstacles moves that end in v_j; if l is the greatest integer $l \geq j$ such that vertex v_l is the origin of either an outward move ending at a vertex $w < v_j$, or a backward move ending at a vertex $w < v_r$ (if A contains no such a move then $l = j$); if f is the greatest integer $j < f \leq l$ such that v_f is the destination of a forward move of $L(A, j)$ (if $L(A, j)$ contains no such forword move then $f = j$); if r is the greatest integer $0 < r < j$ such that the sidestep of v_r is visited by a robot before arriving at v_j (if in A the robot took no sidestep before arriving at v_j then $r = 0$).

A left part $L = L(A, j)$ of class $\mathcal{C}(j, k, l, f, r)$ can be seen as a partial plan that takes the robot from s to v_j but has not decided about the final destinations of k "suspended" obstacles that are temporarily moved by L from their origins (vertices of $\{v_0, v_1, \ldots, v_j\}$) to v_j. The final destinations of the suspended obstacles are going to be specified by the right part $R = R(A, j)$. However, by looking only at the left part of a plan, we can say that a suspended obstacle originally at u can be moved by the right part of the plan only in the following two possible ways:

1. outward to some B_i with $i > j$. In fact, if it is moved to some vertex $w \in B_i$ with $i \leq j$ the move $u \to w$ would have been part of the left part of A.
2. first forward to a vertex v_z, with $z > l$, and then backward to a vertex w such that $v_r \leq w$.

Indeed if it is moved to v_z with $z \leq l$ then the forward move $u \to v_z$ would have been in the left part of A. Similarly, if the obstacle is moved backward to $w < v_r$ then the move would have been in the left part.

For each left part L relative to v_j our algorithm computes the minimum legal completion of L in terms of the minimal legal completions of left parts relative to v_{j+1}.

DEFINITION 4. Let $L_1 \in \mathcal{C}(j, k_1, l_1, f_1, r_1)$ and $L_2 \in \mathcal{C}(j+1, k_2, l_2, f_2, r_2)$. Then, we say that L_1 and L_2 are adjacent iff there exists a LCP A such that $L_1 = L(A, j)$ and $L_2 = L(A, j+1)$.

The difference $M = \Delta(L_1, L_2)$ of two adjacent left parts L_1 and L_2 is the multiset of moves defined as follows: for each move $u \to v \in L_2$ if L_1 contains a move $u \to v'$ then M contains the move $v' \to v$; else M contains the move $u \to v$.

If M is the difference of two adjacent left parts L_1 and L_2 we also say that M is an extension of L_1 to L_2.

LEMMA 2. *The difference $M = \Delta(L_1, L_2)$ of two adjacent left parts $L_1 \in \mathcal{C}(j, k_1, l_1, f_1, r_1)$ and $L_2 \in \mathcal{C}(j+1, k_2, l_2, f_2, r_2)$ can only contain moves of the following types*:

1. *a robot move from v_j to v_{j+1}*;
2. *obstacle moves from v_j to v_{j+1}* (*moves of the suspended obstacles*);
3. *outward moves to vertices of B_{j+1}.*
4. *If v_j is a branch vertex then M may contain robot moves from v_j to its sidestep and back.*

(*a*) *If it does so then $r_2 = j$ and M contains backward moves $u \xrightarrow{j} w$, with $v_{l_1+1} \leq u \leq v_{l_2}$ and $v_r \leq w < v_j$.*

(*b*) *If there is no hole among vertices $v_{f_1+1}, \ldots, v_{l_1}$ then M contains a forward move $v_j \to w$ for each hole w between v_{l_1} and v_{f_2}.*

Proof. Claim 1 is obvious.

The moves mentioned in Claim 2 are the moves relative to suspended obstacles of L_1 that are also suspended in L_2 and, possibly, the move of the obstacle located at v_j.

To prove Claim 3, we observe that outward moves to vertices of B_i with $i < j$ belong to L_1 and L_2 and outward moves to vertices of B_i with $i \geq j+2$ do not belong to L_1 or to L_2 but to their

corresponding right parts. On the other hand, moves to vertices of B_{j+1} do belong to L_2 but not to L_1 and thus they are in M.

To prove claim 4 we first observe that if v_j is not a branch vertex then M does not contain forward and backward moves. In fact, M contains only backward moves ending at vertices between v_{r_1} and v_{r_2}: since in this case $r_2 = r_1$ then M contains no backward move. On the other hand, a left part contains only forward moves of obstacles that are moved backward during the same left part. Since L_1 and L_2 contain the same backward moves they have also the same forward moves.

Suppose now that v_j is a branch vertex and plan A moves the robot to the sidestep of v_j. Thus, M contains the robot moves from v_j to its sidestep and back. To prove Claim 4a we observe that in this case we have $r_2 = j$ and thus M contains all the backward moves ending at vertices $v_{r_1}, v_{r_1-1}, \ldots, v_{j-1}$. These moves are originated in vertices of P between v_{l_1} and v_{l_2}: in fact, for each vertex w a backward move originating at w belongs to both L_1 and L_2 if $w \le v_{l_1}$, and to their right part if $v_{l_2} < w$. If $v_{l_1} < w \le v_{1_2}$, instead, the backward move belongs to L_2 but not to L_1.

Claim 4b is proved by the following observations. First of all, notice that by property 3 of LCP all the forward moves of L_2 are originated at vertices of $v_0, \ldots, v_j$. By property 6 of LCP if there exists a hole between v_{f_1} and v_{l_1} then all the legal completions of L_1 do not contain any forward move $u \to w$ with $u \le v_j$ and $v_{f_1} < w$. Thus, M does not contain forward moves. Suppose, instead, that all the vertices between v_{f_1} and v_{l_1} are full and let $u \to w$ be a forward move of L_2. If $w \le l_1$ then this move belongs to L_1 too; if $v_{l_1} < w \le v_{l_2}$ then by definition 3 the move $u \to w$ is split into two parts: $u \to v_j$ belonging to L_1 and $v_j \to w$ belonging to the corresponding right part R_1. On the other hand $u \to w$ belongs to L_2. Therefore $v_j \to w$ belongs to M. Finally if $v_{l_2} < w$ then the move belongs to the right parts relative to L_1 and L_2 and thus does not belong to M. ■

We say that the difference M of L_1 and L_2 is a (h_1, h_2, h_3)-extension if M contains h_1 outward moves, h_2 forward moves, and h_3 backward moves.

COROLLARY 1. *If M is a (h_1, h_2, h_3)-extension of L_1 to L_2 then the following conditions hold*:

1. *if there exists a hole v_m such that $f_1 < m \le l_1$ then $h_2 = 0$;*
2. *if v_{j+1} is the unique branch vertex between v_j and v_m then $h_1 = k$.*
3. *if $j + 1 \le f_1$ then all the obstacles moved outward are suspended*;
4. *if $h_2 > 0$ and $v_{f'}$ is the h_2th hole of $v_{l_1+1}, \ldots, v_d$ then there are at most $h_3 - h_2$ vertices of P between v_{l_1} and $v_{f'}$ that belong to O;*

Proof. Claim 1 follows directly from Lemma 2 (4b).

To prove Claim 2 we observe that by the previous property all the suspended obstacles of L must be moved outward. Moreover, by property 4 of LCP they must be moved to vertices of $B_{j+1} \cup \cdots \cup B_{m-1}$. Thus, if v_{j+1} is the unique branch vertex between v_{j+1} and v_m all the suspended obstacles must be moved to B_{j+1}.

Claim 3 follows from the observation that L contains a forward move ending at v_{f_1}. By property 4 of LCP the extensions of L cannot contain outward moves from vertices $v_{l_1+1}, \ldots, v_d$ to B_{j+1}. Thus, all the obstacles moved outward in M are suspended.

To prove Claim 4 suppose by sake of contradiction that M contains a forward move ending at $v_{f'}$ and there are more than $h_3 - h_2$ obstacles placed at vertices between v_{l_1} and $v_{f'}$. Consider now the left part $L_2 \in \mathcal{C}(j+1, k_2, l_2, f_2, j)$ reached by M. Since M has a move originating at $v_{f'}$ we have that $f' \le l_2$. However, since M moves only h_3 obstacles backward there is at least an obstacle placed in a vertex between v_{l_1} and $v_{f'}$ that is not moved backward by M. Thus, there is an obstacle placed at a vertex between v_{j+1} and v_{l_2} that is not moved in L_2. But, this contradicts the hypothesis that L_2 is the left part of a LCP. In fact, by Definition 3 all obstacles placed between v_{j+1} and v_{l_2} are moved in L_2 backward or outward. ■

To compute the best (h_1, h_2, h_3)-extension of a left part $L \in \mathcal{C}(j, k, l, f, r)$ we need to decide which obstacles are moved and where they are moved.

4.1. *Constructing the Minimal (h_1, h_2, h_3)-Extension of a Left Part*

In specifying the construction of the minimal extension, for ease of presentation, we will often say that "one suspended obstacle is moved" without specifying which suspended obstacle is actually moved.

The identity of the suspended obstacle actually moved at the various extensions does not affect the cost of the extension of the resulting plan. However, if we are not careful the resulting plan might not be leftmost canonical. We thus adopt the convention that whenever a suspended obstacle is to be selected, we select the suspended obstacle that in the starting configuration is located at the vertex that is furthest away from the current position of the robot.

Let $L \in \mathcal{C}(j, k, l, f, r)$. In computing the extensions of L we select first the obstacles to be moved outward and then the obstacles to be moved forward and backward. To select which obstacles are moved we distinguish two cases:

1. all vertices $v_{f+1}, \ldots, v_l$ are full;
2. there exists a hole among the vertices $v_{f+1}, \ldots, v_l$.

All vertices $v_{f+1}, \ldots, v_l$ are full. By Lemma 2 the h_1 obstacles moved outward are moved to h_1 holes of B_{j+1} and the minimal extension moves these obstacles to the h_1 holes closest to v_{j+1}. If $h_1 \leq k$ then these obstacles are chosen from among the suspended obstacles. In fact, suppose, by sake of contradiction, that there exists an extension M of L to a left part L' that moves a nonsuspended obstacle from $w > v_l$ outward to a vertex of B_{j+1} while at least one of the suspended obstacles is not assigned a destination and thus remains suspended in L'. Let A be a LCP such that $L = L(A, j)$ and $L' = L(A, j+1)$, and let $u \to u'$ be the move in A of this suspended obstacle. Since the obstacle at u is suspended in $L(A, j)$ then $u \leq v_j < u'$. On the other hand, the obstacle at $w > v_j$ is moved to a vertex of B_{j+1} and thus (u, w) is a 5-problematic pair for A. But this contradicts the hypothesis that A is a LCP.

If $h_1 > k$ then, after having moved all k suspended obstacles outward to B_{j+1}, we still need to select $h_1 - k$ obstacles to be moved outward. These obstacles are necessarily taken from vertices $v_{l+1}, \ldots, v_d$. However, observe that if $f \geq j+1$ then by Corollary 1 extensions of L cannot contain outward moves from vertices of $v_{l+1}, \ldots, v_d$ to B_{j+1}. Thus, L does not admit a (h_1, h_2, h_3)-extension with $h_1 > k$ when $f \geq j+1$. If $f = j$, instead, the remaining obstacles to be moved outward are taken from the first $h_1 - k$ full vertices between v_{l+1} and v_d (in left to right order).

Suppose now that the obstacles to be moved outward have already been selected and let k' be the number of obstacles still suspended and l' be the largest integer such that the obstacle at $v_{l'}$ has been moved to B_{j+1} (if only suspended obstacles were moved outward $l' = l$). It remains to consider how to select the h_2 forward moves and the h_3 backward moves of the minimal (h_1, h_2, h_3)-extension.

By Lemma 2 an extension can move forward only suspended obstacles. Thus, L admits no (h_1, h_2, h_3)-extension with $h_1 + h_2 > k$. If $h_1 + h_2 \leq k$ (and thus $l' = l$), instead, by property 4 of LCP the h_2 suspended obstacles moved forward are moved to the h_2 first holes of $v_{l+1}, \ldots, v_d$.

The h_3 obstacles moved backward are located at vertices of $v_{l'+1}, \ldots, v_d$ (recall that for each $j < m \leq l'$ the move originating at v_m has already been considered: if $m \leq l$ it belongs to L; otherwise it is one of the outward moves computed in the previous step). By property 6 of LCP they are taken from the first h_3 vertices of $v_{l'+1}, \ldots, v_d$ that contain obstacles (and thus h_2 of them are obstacles previously moved forward); by property 2, these obstacles are moved to $v_{j-h_3}, \ldots, v_{j-1}$. Therefore, the following algorithm can be used to select the h_2 forward moves and the h_3 backward moves of the extension. Let f' be the index of the h_2th vertex of the path from $v_{l'+1}$ to v_d that is a hole. Scan the vertices $v_{l'+m}$, for $m = 1, 2, \ldots, d - l'$, until you find h_3 obstacles to be moved backward. If $v_{l'+m} \in O$ then add the backward move $v_{l'+m} \overset{j}{\to} v_{j-m}$ to the extension; if $v_{l'+m}$ is a hole and $l' + m \leq f'$, add the forward move $v_j \to v_{l'+m}$ and the backward move $v_{l'+m} \overset{j}{\to} v_{j-m}$.

At least one of the vertices of $v_{f+1}, \ldots, v_l$ is a hole. Let v_m be the hole in $v_{f+1}, \ldots, v_l$ that is closest to v_f. By Corollary 1 extensions of L cannot contain forward moves and outward moves of nonsuspended obstacles. Moreover, if v_{j+1} is the unique branch vertex between v_{j+1} and v_m, then all the suspended obstacles must be moved to B_{j+1}. Thus, L admits only (h_1, h_2, h_3)-extensions with $h_2 = 0$ and $h_1 \leq k$ ($h_1 = k$ if v_{j+1} is the unique branch vertex between v_{j+1} and v_m). The selection of the obstacles that are moved is performed as in Case 1.

From the discussion above, it is clear that the minimal cost (h_1, h_2, h_3)-extension of a left part $L \in \mathcal{C}(j, k, l, f, r)$ can be computed from the knowledge of the parameters j, k, l, f, and r and the location of obstacles in $\mathcal{S}$.

All left parts of $\mathcal{C}(j, k, l, f, r)$ have the same minimal (h_1, h_2, h_3)-extension and thus they all have the same minimum legal completion. In the following we say that class C_1 *reaches* class C_2 by the extension M if and only if there is a left part $L_1 \in C_1$ and a left part $L_2 \in C_2$ such that M is the difference between L_1 and L_2.

4.2. *The Graph $G(\mathcal{S})$*

With this in mind, we construct for each instance $\mathcal{S}$ a weighted directed graph $G(\mathcal{S})$ that has a vertex $w(j, k, l, f, r)$ for each nonempty class $\mathcal{C}(j, k, l, f, r)$ and two distinguished vertices w_s and w_t that we call the source and sink of the graph, respectively. The source w_s is connected to $w(0, 0, 0, 0, 0)$ by an arc with empty label and weight 0; all the vertices $w(d, 0, d, d, r)$ are connected to w_t by arcs with empty label and weight 0. Moreover, there is an arc from $w(j, k, l, f, r)$ to $w(j', k', l', f', r')$ if and only if $\mathcal{C}(j', k', l', f', r')$ is reachable from $\mathcal{C}(j, k, l, f, r)$. This arc is labelled with a compact coding of the minimum cost extension from $\mathcal{C}(j, k, l, f, r)$ to $\mathcal{C}(j', k', l', f', r')$ (i.e., if the minimum cost extension is a (h_1, h_2, h_3)-extension then the arc is labeled with the triplet (h_1, h_2, h_3)) and its weight is set equal to the cost of this extension.

As by the definition of extension there exists no arc in $G(\mathcal{S})$ connecting vertices $w(j, k, l, f, r)$ and $w(j', k', l', f', r')$ with $j' \neq j + 1$; $G(\mathcal{S})$ is a directed acyclic graph (a DAG).

We observe that each path from the source to the sink of $G(\mathcal{S})$ naturally defines a LCP for $\mathcal{S}$, obtained by considering the moves of the extensions that label the arcs of the path. We only have to be careful with regard to the moves of the suspended obstacles as they are broken into various pieces in different extensions. It is thus necessary to concatenate all moves of the same suspended obstacle into a single (outward or forward) move. The cost of the plan is given by the sum of the weights of the arcs of the path.

On the other hand, each LCP A is associated with a path of $G(\mathcal{S})$. In fact, for each $0 \leq j \leq d$, denote by C_j the class such that $L(A, j) \in C_j$; there exists a path in $G(\mathcal{S})$ that visits the vertices corresponding to the classes C_j. The cost of this path is equal to the cost of the best LCP among the plans A^* such that $L(A^*, j) \in C_j$, for each j.

Thus, our problem is reduced to the computation of the shortest path from w_s to w_t in the weighted DAG $G(\mathcal{S})$. The problem can be solved in time proportional to the number of its arcs and vertices. Next we first show how to compute the minimum (h_1, h_2, h_3)-extension of a class, for each h_1, h_2, and h_3, and then we prove that $G(\mathcal{S})$ has $O(d^5)$ vertices and $O(d^7)$ arcs.

To compute the minimum (h_1, h_2, h_3)-extension we use the functions *out*(j, k, l, f, h) and back $(j, k, l, f, r, h_1, h_2)$ defined as follows. *out*(j, k, l, f, h) is defined as the minimum cost of moving h obstacles outward in an extension of a left part of $\mathcal{C}(j, k, l, f, r)$ (actually, this is independent from r); *back*$(j, k, l, f, r, h_1, h_2)$ is defined as the minimum cost of moving h_1 obstacles forward and h_2 obstacles backward in an extension of $\mathcal{C}(j, k, l, f, r)$. Both functions return the actual minimal cost c of performing the moves, the number k^* of remaining suspended obstacles, the greatest index f^* such that a suspended obstacle is moved forward to v_{f^*}, and the greatest index l^* such that an obstacle has been moved from v_{l^*} to $u < v_{j+1}$.

The values of *back* and *out* are computed using the table of the positions of the obstacles in $\mathcal{S}$. This table can be precomputed in time $O(n)$. The computation of the cost returned by *out* is carried out in the following way. Obviously, *out*$(j, k, l, f, 0) = 0$. The value of *out*$(j, k, l, f, h_1 + 1)$ is computed in constant time from the value of *out*(j, k, l, f, h_1) and the length of the path traversed by the obstacle moved to the $(h + 1)$st closest hole of B_{j+1}. The origin of this move is computed as described in Section 4.1. We can thus conclude that, for each j, k, l, and f, the values of *out*(j, k, l, f, h_1) for all needed h_1's can be computed in time $O(\min\{d, |\mathrm{B}_{j+1}|\}))$.

Similarly, for each j, k, l, f, and r all the needed values of *back*(j, k, l, r, h_1, h_2) are computed in an incremental way in time $O((\min\{d, |\mathrm{Ch}(r, j)|\})^2)$.

In Fig. 13 we describe algorithm EXTEND that computes the minimum cost (h_1, h_2, h_3)-extension M of a class $\mathcal{C}(j, k, l, f, r)$ and the class reached from $\mathcal{C}(j, k, l, f, r)$ by M. Assuming that all needed values of *out* and *back* have been computed in a preprocessing phase, the running time of EXTEND is clearly $O(1)$.

The next lemma proves that algorithm EXTEND can be used to compute the labels and the weights of the arcs directed out from a vertex $w(j, k, l, f, r)$ of $G(\mathcal{S})$ in time $O(d^3)$.

EXTEND$(j, k, l, f, r, h_1, h_2, h_3)$
Input: A 5-tuple (j, k, l, f, r) describing a class of left parts and three integers h_1, h_2 and h_3.
begin
  $(c_{h_1}, k_{h_1}, l_{h_1}, f_{h_1}) \leftarrow out(j, k, l, f, h_1)$;
  $(c_{h_2,h_3}, k_{h_2,h_3}, l_{h_2,h_3}, f_{h_2,h_3}) \leftarrow back(j, k_{h_1}, l_{h_1}, f_{h_1}, r, h_2, h_3)$;
  $c \leftarrow c_{h_1} + c_{h_2,h_3} + (k - k_{h_2,h_3} + 1)$;
  if $(h_3 > 0)$ **then** $r_{h_2,h_3} \leftarrow j$ **else** $r_{h_2,r_3} \leftarrow r$;
  return $(c, \mathcal{C}(j+1, k_{h_2,h_3}, l_{h_2,h_3}, f_{h_2,h_3}, r_{h_2,h_3}))$;
end

FIG. 13. The algorithm EXTEND.

LEMMA 3. *For each class $\mathcal{C}(j, k, l, f, r)$ it is possible to compute all the minimal cost extensions of $\mathcal{C}(j, k, l, f, r)$ in time $O(d^2)$, if the vertices $v_{f+1}, \ldots, v_l$ are full, and in time $O(d^3)$ otherwise.*

Proof. By Lemma 2 it follows that an extension of $\mathcal{C}(j, k, l, f, r)$ moves at most $\min\{d, |\mathrm{Ch}(r, j)| + |\mathrm{B}_{j+1}|\}$ obstacles to vertices of B_{j+1} and $\mathrm{Ch}(r, j)$. These obstacles can be moved outward or backward and for each possible number of obstacles moved outward and backward we have a different extension. Moreover, if there is no hole between v_f and v_l obstacles moved backward can be either suspended obstacles, previously moved forward, or obstacles taken from $v_{l+1}, \ldots, v_d$. Thus, if no forward move has to be considered, we need to compute $O(d^2)$ distinct extension of $\mathcal{C}(j, k, l, f, r)$; if, instead, forward moves also have to be considered then we have to compute $O(d^3)$ distinct extensions. In both cases the costs of all these extensions, together with the classes reached, can be computed in time linear in the number of extensions using functions *out* and *back*. ■

THEOREM 3. *An optimal LCP for an instance $\mathcal{S}$ of STMP1R can be obtained by solving a shortest path problem in a weighted DAG with $O(d^7)$ arcs and $O(d^5)$ vertices.*

Proof. By the previous discussion we have that there is a plan for $\mathcal{S}$ if and only if there exists a path between the source and the sink of the weighted DAG $G(\mathcal{S})$. Moreover, the optimal plan for $\mathcal{S}$ is obtained by the shortest source-sink path of $G(\mathcal{S})$ by taking the moves of the labels associated to the arcs of the path.

It is easy to see that $G(S)$ has $O(d^5)$ vertices. By Lemma 3 there are $O(d^4)$ vertices of $G(\mathcal{S})$ that have $O(d^3)$ outgoing arcs, while all the remaining vertices have $O(d^2)$ outgoing arcs, giving a total of $O(d^7)$ arcs.

COROLLARY 2. *There exists an algorithm that, for each instance of the STMP1R that admits a solution of finite cost, computes the cost of an optimal plan in time $O(n + d^7)$.*

Proof. The graph $G(\mathcal{S})$ can be constructed in time $O(n + d^7)$ and the shortest path problem can be solved in time $O(d^7)$.

5. A MORE EFFICIENT ALGORITHM FOR STMP1R

The algorithm given in the previous section solves an instance $\mathcal{S}$ of STMP1R by constructing a weighted directed acyclic graph $G(\mathcal{S})$ with $O(d^5)$ vertices and $O(d^7)$ arcs and solving a shortest path problem on $G(\mathcal{S})$. In this section we give the construction of a directed acyclic graph $H(\mathcal{S})$ that has $O(d^5)$ vertices, $O(d^4 \cdot \min\{n, d^2\})$ arcs, and two distinguished vertices called source and sink, and still encodes the STMP1R instance $\mathcal{S}$ so that a path from the source to the sink in this graph corresponds to a LCP for $\mathcal{S}$ and the shortest of such paths corresponds to an optimal plan for $\mathcal{S}$. We shall prove that $H(\mathcal{S})$ can be constructed in time $O(n + d^4 \cdot \min\{n, d^2\})$.

We will obtain $H(\mathcal{S})$ in two steps. We first show how to construct a graph $J(\mathcal{S})$ that has $O(d^4 \min\{n, d^2\} + d^6\})$ arcs and still encodes the STMP1R instance $\mathcal{S}$. Then, we show how $H(\mathcal{S})$ can be obtained from $J(\mathcal{S})$.

5.1. *The Graph $J(\mathcal{S})$*

The following observation is at the base of the construction of $J(\mathcal{S})$. Each extension M of the left part L can be thought of as consisting of two parts: the *out extension*, which includes all the outward

moves, and the *back extension*, which includes all the other moves (robot moves, forward and backward moves of obstacles, and moves of obstacles that remain suspended). More formally, let A be a LCP and let M be the (h_1, h_2, h_3)-extension of $L(A, j)$ to $L(A, j+1)$. We partition the moves of M in a h_1-out extension M_{out} and a (h_2, h_3)-back extension M_{back} according to the following rule:

$$u \xrightarrow{z} w \text{ belongs to } \begin{cases} M_{\text{out}} & \text{if } z \neq R \text{ (the moved object is an obstacle) and } w \in B_{j+1}; \\ M_{\text{back}} & \text{otherwise.} \end{cases}$$

This partitioning of M into two parts corresponds to insert a new vertex on the arc of $G(\mathcal{S})$ labelled with M, thus splitting it in two arcs: the first corresponding to the out extension and the second to the back extension. This leads to a partitioning of a LCP that is a refinement of the partition presented in the previous section. Plan A is partitioned with respect to a vertex $v_j \in P$ and a bit b: the left part $L(A, j, 0)$ of A with respect to v_j and 0 is defined as equal to $L(A, j)$; the left part $L(A, j, 1)$ of A with respect to v_j, and 1 is defined as consisting of all the moves of $L(A, j)$ and all the moves of M_{out}. As before, the set of all the possible left parts of LCP for $\mathcal{S}$ relative to a vertex v_j and b can be partitioned in classes according to four parameters k, l, f, and r.

DEFINITION 5. *For any LCP A, the left part $L(A, j, b)$ belongs to the class $\mathcal{C}(j, k, l, f, r, b)$ if and only if the following conditions hold*:

1. *there are k suspended obstacles at v_j;*
2. *r is the greatest integer, with $0 < r < j$, such that the sidestep of v_r is visited by the robot before arriving at v_j ($r = 0$ if the robot does not visit any sidestep vertex before arriving at v_j).*
3. *l is the greatest integer, with $j \leq l \leq d$, such that vertex v_l is the origin of either an outward move ending at a vertex $w < v_{j+1}$ or a backward move ending at a vertex of* $\text{Ch}(r, 0)$ *(if A contains no such move then $l = j$);*
4. *f is the greatest integer, with $j \leq f \leq l$, such that vertex v_f is the destination of a forward move and the origin of a backward move to a vertex of* $\text{Ch}(r, 0)$ *(if $L(A, j)$ contains no such move then $f = j$);*

As done before for $G(\mathcal{S})$, we let the graph $J(\mathcal{S})$ have one vertex $u(j, k, l, f, r, b)$ for each nonempty class $\mathcal{C}(j, k, l, f, r, b)$ and two distinguished vertices u_s and u_t, called source and sink of the graph, respectively. Moreover, $J(\mathcal{S})$ has an arc from $u(j_1, k_1, l_1, f_1, r_1, b_1)$ to $u(j_2, k_2, l_2, f_2, r_2, b_2)$ if and only if either $b_1 = 0$ and $b_2 = 1$ and there exists an out-extension M_{out} that extends class $\mathcal{C}(j_1, k_1, l_1, f_1, r_1, b_1)$ into class $\mathcal{C}(j_2, k_2, l_2, f_2, r_2, b_2)$; or $b_1 = 1$ and $b_2 = 0$ and there exists a back-extension M_{back} that extends class $\mathcal{C}(j_1, k_1, l_1, f_1, r_1, b_1)$ into class $\mathcal{C}(j_2, k_2, l_2, f_2, r_2, b_2)$. In both cases the arc is labelled with the extension and has weight equal to the cost of the extension. Moreover, there are zero-weight arcs from u_s to $u(0, 0, 0, 0, 0, 0)$ and, for each r, from $u(d, 0, d, d, r, 0)$ to u_t.

The graph $J(\mathcal{S})$ is still a DAG with $O(d^5)$ vertices. The vertex $u(j, k, l, f, r, 0)$ has out-degree at most $\min\{d, |\text{B}_{j+1}|\}$ (no more than $|\text{B}_{j+1}|$ obstacles can be moved outward to B_{j+1} and in any case no more than d obstacles are moved); vertex $u(j, k, l, f, r, 1)$, instead, has out-degree at most $\min\{d, |\text{Ch}(r, j)|\}$, if there exists a hole between v_f and v_l (no more than $|\text{Ch}(r, j)|$ obstacles can be moved backward to $\text{Ch}(r, j)$ and these obstacles are taken from vertices $v_{l+1}, \ldots, v_d$ that are full), and out-degree $\min\{d, |\text{Ch}(r, j)|\}^2$ otherwise (as in the previous case no more than $\min\{d, |\text{Ch}(r, j)|\}$ obstacles can be moved backward but these obstacles can be taken either from holes where previous forward moves have taken the obstacles or from full vertices). It can be easily seen that the number of arcs of $J(\mathcal{S})$ is $O(d^4 \cdot \min\{n, d^2\} + d^6) = O(d^6)$. The computation of the shortest path from u_s to u_t, and thus of the minimum cost LCP for $\mathcal{S}$, can be carried out in time $O(d^6)$.

5.2. *The Graph $H(\mathcal{S})$*

The number of arcs of the graph that encodes the STMP1R instance $\mathcal{S}$ can be further reduced by observing that some back extensions can be ignored in computing the optimal plan for $\mathcal{S}$. Consider two left parts L_r and L_z belonging to the classes $\mathcal{C}(j, k, l, f, r, 1)$ and $\mathcal{C}(j, k, l, f, z, 1)$, respectively, where v_z is a branch vertex between v_r and v_j. Moreover, let M_r and M_z be the minimal (h_1, h_2)-back extensions of L_r and L_z, respectively. Then the following observation is easy to prove.

Observation. If $1 \le h_2 \le |\mathrm{Ch}(z, j)|$, M_r and M_z are equal (they contain the same moves) and extend L_r and L_z to the same class $\mathcal{C}(j+1, k', l',\ f',\ j, 0)$, for some k'. l' and f'.

Based on the above observation, we consider the graph $H(\mathcal{S})$ obtained from $J(\mathcal{S})$ in the following way.

The set of vertices of $H(\mathcal{S})$ consists of

- two distinguished vertices called source and sink of the graph;
- a vertex $u(j, k, l,\ f, r, 0)$ for each $\mathcal{C}(j, k, l,\ f, r, 0)$;
- a vertex $u(j, k, l,\ f, r, 1)$ for each class $\mathcal{C}(j, k, l,\ f, r, 1)$ such that v_j is not a branch vertex; two vertices $u(j, k, l,\ f, r, 1)$ and $u^*(j, k, l,\ f, r, 1)$ for each class $\mathcal{C}(j, k, l,\ f, r, 1)$ such that v_j is a branch vertex.

Let $C = \mathcal{C}(j, k, l,\ f, r, 1)$ and let M_{h_1,h_2} be the minimal (h_1, h_2)-back extension of C that extends C into class $C_{h_1,h_2} = \mathcal{C}(j+1, k', l',\ f', r', 0)$. Vertex $u(j, k, l,\ f, r, 1)$ has one outgoing arc to vertex $u(j+1, k_{0,0}, l_{0,0},\ f_{0,0}, r_{0,0}, 0)$ of weight equal to the cost of $M_{0,0}$ and one of zero weight going to $u^*(j, k, l,\ f, r, 1)$, if this vertex exists. Vertex $u^*(j, k, l,\ f, r, 1)$, if it exists, has one outgoing arc for each (h_1, h_2)-back extension M_{h_1,h_2}, for $h_2 > |\mathrm{Ch}(j, z)|$, of weight equal to the cost of M_{h_1,h_2} and one arc of weight 0 going to vertex $u(j, k, l,\ f, z, 1)$, where z is the smallest integer $r < z < j$ such that v_z is a branch vertex. Vertex $u(j, k, l,\ f, r, 0)$ has one outgoing arc for each out extension. The arc corresponding to the h-out extension has weight equal to the cost of the minimal h-out extension and reaches vertex $u(j, k_h, l_h,\ f, r, 1)$ if class $\mathcal{C}(j, k, l,\ f, r, 0)$ is extended into class $\mathcal{C}(j, k_h, l_h,\ f, r, 1)$ by the minimal h-out extension. The arcs outgoing from the source vertex and ingoing to the sink vertices are defined as in $J(\mathcal{S})$.

The rationale behind the construction of $H(\mathcal{S})$ is that, by Observation 1, not all (h_1, h_2)-back extensions of class $\mathcal{C}(j, k, l,\ f, r, 1)$ are explicitly represented by one arc. Indeed, let $v_{z_0}, v_{z_1}, \ldots, v_{z_s}$ be the branch vertices between $v_r = v_{z_0}$ and $v_j = v_{z_s}$ and assume that $|\mathrm{Ch}(j, z_{s^*+1})| < h_2 \le |\mathrm{Ch}(j, z_{s^*})|$ for some z_{s^*}. Then, by the observation above the minimal (h_1, h_2)-back extension of $\mathcal{C}(j, k, l,\ f, r, 1)$ is equal to the minimal (h_1, h_2)-back extension of $\mathcal{C}(j, k, l,\ f, z_{s^*}, 1)$. The (h_1, h_2)-back extension of class $\mathcal{C}(j, k, l,\ f, r, 1)$ is implicitly represented in $H(\mathcal{S})$ by the path consisting of the arc from $u(j, k, l,\ f, r, 1)$ to $u^*(j, k, l,\ f, r, 1)$, the arcs from $u^*(j, k, l,\ f, z_i, 1)$ to $u^*(j, k, l,\ f, z_{i+1}, 1)$, for $i = 0, 1, \ldots, s^*$, and the arc corresponding to the (h_1, h_2)-back extension of $u(j, k, l,\ f, z_{s^*}, 1)$ (See Fig. 14). Therefore each

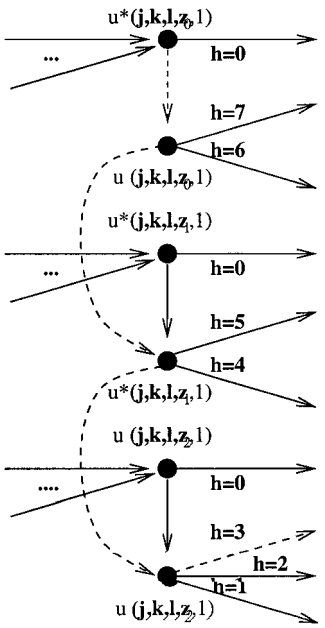

FIG. 14. Vertices $H(\mathcal{S})$ corresponding to some class relative to branch vertices $v_{z_0}, v_{z_1}, v_{z_2}$ with $v_{z_3} = v_j$ are shown. $\mathrm{Ch}(j, z_0)$, $\mathrm{Ch}(j, z_1)$, and $\mathrm{Ch}(j, z_2)$ have size 7, 5, and 3, respectively. The dashed path corresponds to the 3-back extension of class $\mathcal{C}(j, k, l,\ f, z_0, 1)$.

path from source to destination still encodes a LCP and the weight of the shortest of such paths is equal to the cost of the minimal LCP.

We shall prove that $H(\mathcal{S})$ is DAG with $O(d^4 \cdot \min\{n, d^2\})$ arcs and that it can be constructed in time $O(n + d^4 \cdot \min\{n, d^2\})$. Thus, we can conclude that there is an $O(n + d^4 \cdot \min\{n, d^2\})$ algorithm that, for each instance $\mathcal{S}$ of STMP1R that admits a finite cost solution, computes a minimal cost LCP for $\mathcal{S}$.

LEMMA 4. *For each instance $\mathcal{S}$ of STMP1R, the graph $H(\mathcal{S})$ is acyclic.*

Proof. We first observe that, since the source u_s has indegree 0 and the sink u_t has outdegree 0, there is no cycle in $H(\mathcal{S})$ containing u_s or u_t. Thus, we have only to show that there is no cycle in $H(\mathcal{S})$ consisting of vertices distinct from u_s and u_t. To this aim we show that there exists a partial ordering $\leq_{\mathcal{S}}$ on the classes $\mathcal{C}(j, k, l, f, r, b)$ such that for each arc $(u(j, k, l, f, r, b), u(j', k', l', f', r', b'))$ in $H(\mathcal{S})$ it holds that $\mathcal{C}(j, k, l, f, r, b) \leq_{\mathcal{S}} \mathcal{C}(j', k', l', f', r', b')$. Let $\mathcal{C}(j, k, l, f, r, b) \leq_{\mathcal{S}} \mathcal{C}(j', k', l', f', r', b')$ if and only if

$$\begin{array}{lll} \mathcal{C}(j, k, l, f, r, b) = \mathcal{C}(j', k', l', f', r', b') & & \text{or} \\ j < j' & & \text{or} \\ j = j' & \text{and } b < b' & \text{or} \\ j = j' & \text{and } b = b' \text{ and} & r > r'. \end{array}$$

Consider, now the arcs outgoing from vertex $u(j, k, l, f, r, b)$: if $b = 0$, each of these arcs is directed to a vertex $u(j, k', l', f, r, 1)$, where $\mathcal{C}(j, k, l, f, r, 0) \leq_{\mathcal{S}} \mathcal{C}(j, k', l', f, r, 1)$; if $b = 1$, instead, each arc is directed to either $u(j + 1, k', l', f', r', 0)$ or to $u^*(j, k, l, f, z, 1)$, where $r < z < j$. In both cases the vertex reached by the arc corresponds to a class that follows $\mathcal{C}(j, k, l, f, r, 1)$ in the ordering $\leq_{\mathcal{S}}$. ■

LEMMA 5. *The weighted DAG $H(\mathcal{S})$ associated with the instance $\mathcal{S}$ of STMP1R has $O(d^4 \cdot \min\{n, d^4\})$ arcs.*

Proof. Let us count the number of arcs going out of vertices of $H(\mathcal{S})$.

The vertices u_s and $u(d, k, l, f, r, 0)$ have at most one outgoing arc. Vertex $u(j, k, l, f, r, 0)$, for $j < d$, has at most $\min\{d, |B_{j+1}|\}$ outgoing arcs, corresponding to the out extensions of $\mathcal{C}(j, k, l, f, r, 0)$. Vertex $u(j, k, l, f, r, 1)$ has at most two outgoing arcs, one corresponding to the (0.0)-back extension and another directed to $u^*(j, k, l, f, r, 1)$, if this vertex exists.

The computation of the arcs outgoing from $u^*(j, k, l, f, r, 1)$ is a little more involved since we have to distinguish between vertices corresponding to classes that can move obstacles forward in their back extensions and vertices corresponding to classes that cannot have forward moves in their extensions. Let z be the minimum integer $r < z \leq j$ such that v_z is a branch vertex. If there is a hole in P between v_f and v_l then vertex $u^*(j, k, l, f, r, 1)$ has at most $1 + \min\{d - |\mathrm{Ch}(z, j)|, |\mathrm{Ch}(r, z)|\}$ outgoing arcs, one for each $(0, h_2)$-back extension of $\mathcal{C}(j, k, l, f, r, 1)$ with $h_2 > |\mathrm{Ch}(z, j)|$, and one arc of weight 0 directed to $u^*(j, k, l, f, z, 1)$. If, instead, all vertices of P between v_f and v_l are full then vertex $u^*(j, k, l, f, r, 1)$ has at most $1 + \min\{d - |\mathrm{Ch}(z, j)|, |\mathrm{Ch}(r, z)|\}^2$ outgoing arcs, one for each back (h_1, h_2)-back extension of $\mathcal{C}(j, k, l, f, r, 1)$ for $h_2 > |\mathrm{Ch}(z, j)|$ and $h_1 \leq h_2$, and one arc of weight 0 directed to $u^*(j, k, l, f, z, 1)$.

Summing over all the vertices of $H(\mathcal{S})$ we obtain that the number of arcs is

$$\begin{aligned} |E(H(\mathcal{S})| &= \sum_{j=0}^{d-1} \sum_{k,l,f,r} O(\min\{d, |B_{j+1}|\}) + \sum_{j=0}^{d-1} \left(\sum_{k,l,f} O(1 + \min\{d - |\mathrm{Ch}(z, j)|, |\mathrm{Ch}(r, z)|\}) \right. \\ &\quad \left. + \sum_{k,l} O(1 + \min\{d - |\mathrm{Ch}(z, j)|, |\mathrm{Ch}(r, z)|\}^2) \right) + O(d) \\ &= O(d^4 \cdot \min\{n, d^2\}). \end{aligned}$$

■

By the previous observations and Lemma 5 we obtain the following theorem.

THEOREM 4. *An optimal LCP for an instance $\mathcal{S}$ of STMP1R can be obtained by solving a shortest path problem in a weighted DAG with $O(d^4 \cdot \min\{n, d^2\})$ arcs and $O(d^5)$ vertices.*

COROLLARY 3. *There exists an algorithm that, for each instance of STMP1R that admits a finite cost solution, computes the cost of an optimal solution in time $O(n + d^4 \cdot \min\{n, d^2\})$.*

Proof. The construction of $H(\mathcal{S})$, once a $O(n)$-time preprocessing is performed, takes time proportional to the number of arcs. The shortest path problem in the weighted DAG $H(\mathcal{S})$ is solved in time $O(d^4 \cdot \min\{n, d^2\})$.

6. SOLVING TMP1R

In this section we extend our algorithm for STMP1R to TMP1R. We address each of the two restrictions imposed on TMP1R to obtain STMP1R separately, starting with the computation of a plan where the robot can perform a backup part. In Section 6.2 we compute a plan that can move obstacles from and to the sidesteps vertices.

6.1. *Plans with Backup Parts*

In this section, we show how to modify the algorithm for STMP1R to consider also plans with a nonempty backup part. We call the problem of computing the minimum cost plan with a possibly non-empty back part and no obstacle moves starting or ending to sidestep vertices the Back-STMP1R.

Denote by s' the vertex of B_0 of degree greater than 2 that is closest to s (possibly s itself): denote also by s_1 and s_2 two neighbors of s' such that $s_1, s_2 \prec s'$ (in the following we call s_1 and s_2 backup vertices). Let P_{back} be the set of vertices visited by the robot during the backup part of a canonical plan and let $H(s_1) = \{w \in B_0 | w \leq s_1\}$. In [4] it is shown that P_{back} consists of all the vertices that are on the path between s_1 and s and possibly s_2. During the backup part the plan moves some obstacles of P_{for} to vertices of B_0. This is achieved according to the following four-step procedure which is executed before the robot starts travelling on P:

Step 1. Move obstacles placed at vertices of P_{back} to vertices of B_0 not in P_{back};

Step 2. Move the robot to s_1 and move some obstacles of P_{for} to vertices of B_0 that belong neither to P_{back} nor to $H(s_1)$;

Step 3. Move the robot at s_2 and move some obstacles of P_{for} to vertices of $H(s_1)$ (if no obstacle is moved to H this step is skipped and the obstacle in s_2 is not moved);

Step 4. Move the robot to s.

It is easy to see the backup part of a LCP satisfies the following properties:

1. it moves the robot only to s or to vertices of B_0;
2. it does not contain outward or forward moves;
3. no backward move ends at a vertex on the path between s and the last backup vertex visited by the robot;
4. there exists an integer l, with $1 \leq l \leq d$, such that an obstacle at v_m is moved backward during the backup part if and only if $1 \leq m \leq l$.

The backup part of a LCP A can be seen as a sort of left part of the plan with respect to $v_0 = s$ and we denote it by $L(A, 0)$. We classify the left parts $L(A, 0)$ into $d + 1$ classes $\mathcal{C}(0), \ldots, \mathcal{C}(d)$, with $\mathcal{C}(l)$ containing the backup parts that move backward all the obstacles placed in the path from s to v_l and denote by D_l the backup part of $\mathcal{C}(l)$ having minimum cost. Notice that the backup parts of $\mathcal{C}(l)$ differ for the destinations of the obstacles they move but they all induce, after their execution, the same distribution of obstacles in $T - B_0$ and leave the vertices between v_l and the last backup vertex visited free of obstacles. Thus, all the backup parts of $\mathcal{C}(l)$ have the same minimal legal completion that we denote by A_l. Therefore, the optimal plan for $\mathcal{S}$ can be computed as the minimum cost plan over l of the plans $D_l \cup A_l$. Next, we shall show how to compute D_l and A_l for all l.

We compute D_l by computing $D_1(l)$, the minimum cost backup part of $\mathcal{C}(l)$ that visits only one backup vertex (s_1), and $D_2(l)$, the minimum cost backup part of $\mathcal{C}(l)$ that visits both the backup vertices. $D_1(l)$ is computed in the following way. For each possible choice of s_1, let $D_1(l, s_1)$ be the minimum cost backup part that uses s_1 as backup vertex and move all the obstacles between s and v_l backward while the robot is at s_1. Then $D_1(l, s)$ includes the following moves:

1. robot moves from s to s_1 and back to s;
2. obstacle moves to clear the path from s to s_1;
3. backward moves of the obstacles of the critical path between s and v_l to vertices of B_0 not in $\mathrm{P}_{back} \cup H(s_1)$.

For each selection of s_1 we can compute the costs of all the plans $D_1(l, s_1)$, for $l = 1, 2, \ldots, d$, in time $O(d)$, if the list of the holes of $H(s_1)$ sorted by distance from s_1 is available (this can be done in time $O(n)$ during the preprocessing stage). Thus, the algorithm to compute the costs of all the plans $D_1(l) = \min_{s_1} D_1(l, s_1)$, for $l = 1, 2 \ldots, d$, takes time $O(dn)$.

The computation of the cost of $D_2(l)$ is a little more involved. In fact, for each possible selection of s_1 the cost of the plan depends on the number l_1 of obstacles moved to $H(s_1)$ and on the value of s_2. We notice that, given s_1 and l_1, the value of s_2 is fixed: in fact, it is the neighbor of s' that is closest to the $l - l_1 + 1$th hole of B_0 not in $\mathrm{P}_{back} \cup H(S_1)$ (this hole can be obtained from the list computed in the preprocessing phase). Therefore, the cost of $D_2(l)$ is equal to the minimum, for all the selections of s_1 and l_1, of the sum of the costs of $D_1(l_1, s_1)$ plus the cost of moving the obstacle at s_2 (if there is any) to the closest hole, and then the cost of moving the robot from s_1 to s_2 and the cost of moving l_1 obstacles to the closest holes of $H(s_1)$.

It can be easily seen that the costs of all the plans $D_2(l)$, for $l = 1, 2, \ldots, d$, can be computed in time $O(nd^2)$. Since D_l is equal to the plan of minimum cost between $D_1(l)$ and $D_2(l)$, we can conclude that $D(0), D(1), \ldots, D(l)$ can be computed in time $O(nd^2)$.

Let us now shift our focus to the problem of computing the minimal forward part for each class $\mathcal{C}(l)$. We notice that $\mathcal{C}(0)$ contains only the empty backup part and its minimal forward part can be found in time $O(n + d^4 \cdot \min\{n, d^2\})$ by searching a shortest path between the source and the sink in $H(\mathcal{S})$ (see Section 5).

For $l > 0$, finding a minimal forward part for $\mathcal{C}(l)$ is equivalent to finding an optimal LCP for the instance $\mathcal{S}' = (O', s', t')$ of STMP1R on the tree T' defined as follows:

- T' is the subtree of T consisting of the vertices of $T - B_0$ plus the vertices of the path from s_2 to s (assume that s_2 is the last backup vertex by the robot during the backup part of the plan);
- $s' = s$ and $t' = t$;
- for each vertex $w \in T'$ we have that $w \in O'$ if and only if $w \in O$ and it does not lie on the path from s_2 to v_l.

In fact, in $\mathcal{S}'$ no vertex behind s has degree greater than 2 and, consequently, all the optimal canonical plans for $\mathcal{S}'$ have empty backup parts. We have thus reduced the problem of computing the minimal forward part of a LCP with backup part D_l to the problem of computing the shortest path between $u(0, 0, l, 0, 0)$ and the sink u_t in the graph $H(\mathcal{S}')$ constructed as shown in Section 5. We remind the reader that vertex $u(0, 0, l, 0, 0)$ corresponds to the left parts of LCP with respect to $s = v_0$, where there are no suspended obstacles, all the obstacles in vertices between v_0 and v_1 are moved backward to B_0, and the robot has taken no sidestep. Moreover, we add to the graph $H(\mathcal{S}')$ the arcs $(u_s, u(0, 0, l, 0, 0))$, for $l = 1, \ldots, d$, having weight equal to the cost of D_l. In this way, it is easy to see that the shortest path form u_s to u_t defines a minimal LCP that has a non-empty backup part for the instance $\mathcal{S}$ of Back-STMP1R. Comparing the plans defined by the shortest source-sink paths in $H(\mathcal{S})$ and $H(\mathcal{S}')$ we obtain the optimal LCP for the instance $\mathcal{S}$ of Back-STMP1R.

6.2. *Sidesteps with Obstacles*

The possibility for a sidestep vertex to have an obstacle raises two kinds of problems.

First, whether a sidestep vertex can be the destination of a move depends both on the type of the move and on whether the robot visits this sidestep. Consider a sidestep vertex v and let v_j be its

corresponding branch vertex. If v is visited by the robot during the execution of plan A then, v cannot be occupied by an obstacle at the moment in which the robot arrives at v_j. This means that v cannot be the destination of an outward move but, on the other hand, v can still be the destination of a forward move and the origin of a backward move that takes place before the robot arrives at v_j. If instead v is not visited by the robot during the execution of A, then v can be the destination of only outward moves.

The second kind of problem occurs when considering those obstacles that, in $\mathcal{S}$, were in the sidesteps visited by the robot. The algorithm has to move these obstacles before the robot comes to the corresponding branch vertices.

We will consider first the case in which in $\mathcal{S}$ all the sidestep vertices of P are holes but obstacles can be moved to (and possibly from) the sidesteps. We then address the case in which sidesteps can host obstacles even in the starting configuration of the problems.

6.2.1. Obstacles at the sidestep vertices. Consider the back extensions of the class $\mathcal{C}(j, k, l, f, r, 1)$. When the robot comes to the sidestep vertex of v_j there is a hole in the sidestep vertex of v_r that can be used by the plan as the destination of a backward move. The fact that sidestep vertices can be destinations of backward moves has to be taken into account in computing back extensions. In particular, we need to consider that there is one more available destination for backward moves. Thus, for each vertex $u(j, k, l, f, r, 1)$ of $H(\mathcal{S})$ we need an arc outgoing from $u(j, k, l, f, r, 1)$ for each back extension of the class $\mathcal{C}(j, k, l, f, r, 1)$. Notice that, if at least a vertex in $v_{f+1}, \ldots, v_l$ is a hole then $\mathcal{C}(j, k, l, f, r, 1)$ has a (h_1, h_2)-back extension for $h_1 = 0$ and $h_2 = 0, 1, \ldots, |\mathrm{Ch}(j, r)| + 1$. However, the arcs corresponding to the back extensions with $h_2 \leq |\mathrm{Ch}(j, r)|$ are already present in $H(\mathcal{S})$. Thus, we have only to add the arc corresponding to the $(0, |\mathrm{Ch}(j, r)| + 1|)$-back extension. If there is no hole in $v_{f+1} \ldots, v_l$, instead, $\mathcal{C}(j, k, l, f, r, 1)$ has a (h_1, h_2)-back extension for $h_1 = 0, 1, \ldots, \min\{k, h_2\}$ and $h_2 = 0, 1, \ldots, |\mathrm{Ch}(j, r)| + 1$. As for the previous case, most of the arcs corresponding to these back extensions are already present in $H(\mathcal{S})$ and thus we have to add $O(d)$ arcs corresponding to the back extensions with $h_2 = |\mathrm{Ch}(j, r)| + 1$. It is easy to see that when adding these arcs to $H(\mathcal{S})$ the graph is still a DAG with $O(d^4 \cdot \min\{n, d^2\})$ arcs.

To take into account the fact that sidestep vertices that do not belong to P_{for} can be the destination of outward moves but not the origin of obstacle moves, while the sidestep vertices of P_{for} can be the origin of obstacle moves and the destination of forward moves, we modify the graph $H(\mathcal{S})$ in the following way (see Figs. 15 and 16 for an example).

Each vertex $u(j, k, l, f, r, 0)$ of $H(\mathcal{S})$ is replaced with two vertices that we call $u_Y(j, k, l, f, r, 0)$ and $u_N(j, k, l, f, r, 0)$, corresponding respectively to the case in which the sidestep of v_j is not the destination of any outward move (and thus the robot has to take a sidestep) and to the case in which the sidestep of v_j is the destination of an outward move (and thus the robot cannot take a sidestep).

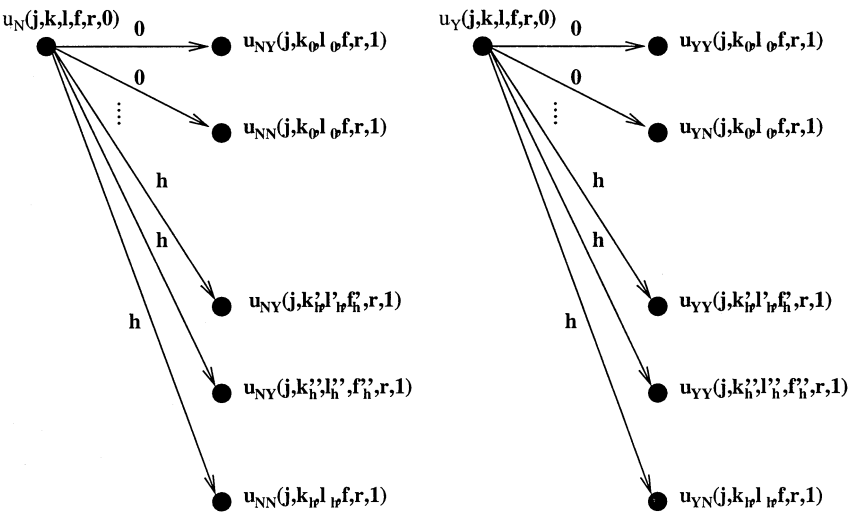

FIG. 15. Arcs outgoing from the vertices $u_Y(j, k, l, f, r, 0)$ and $u_N(j, k, l, f, r, 0)$, corresponding to the class $\mathcal{C}(j, k, l, f, r, 0)$.

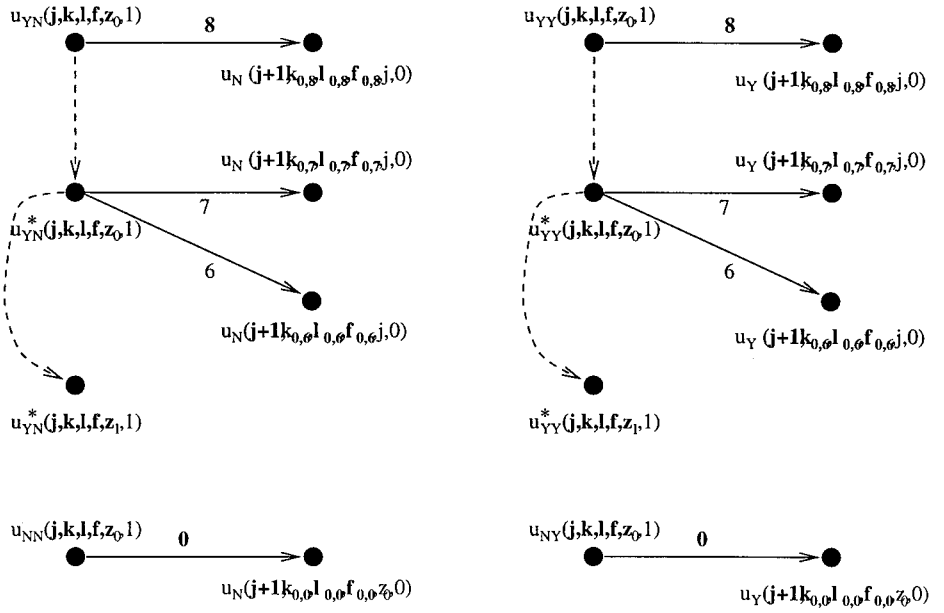

FIG. 16. Arcs outgoing from the vertices $u_{YY}(j,k,l,\ f,z_0,1), u_{YN}(j,k,l,\ f,z_0,1), u_{NY}(j,k,l,\ f,z_0,1)$, and $u_{NN}(j,k,l,\ f,z_0,1)$, corresponding to vertex $u(j,k,l,\ f,z_0,1)$ of Fig. 14. Each arc corresponds to a back-extension of the class $\mathcal{C}(j,k,l,\ f,z_0)$. We are supposing that there is a hole in P between v_f and v_l and thus all the back-extensions do not contain forward moves.

Vertices $u^*(j,k,l,\ f,r,1)$ and $u(j,k,l,\ f,r,1)$, instead, are both replaced with four vertices. We discuss only the transformation for $u(j,k,l,\ f,r,1)$ as the transformation for $u^*(j,k,l,\ f,r,1)$ is exactly the same. Vertex $u(j,k,l,\ f,r,1)$ of $H(\mathcal{S})$ is replaced by vertices $u_{YY}(j,k,l,\ f,r,1)$, $u_{YN}(j,k,l,\ f,r,1)$, $u_{NY}(j,k,l,\ f,r,1)$, and $u_{NN}(j,k,l,\ f,r,1)$ with the following intended meaning: vertex $u_{YY}(j,k,l,\ f,r,1)$ corresponds to situations in which the robot takes a sidestep both at v_j and v_{j+1}; vertex $u_{NN}(j,k,l,\ f,r,1)$ corresponds to situations in which the robot sidesteps neither at v_j nor at v_{j+1}; vertex $u_{YN}(j,k,l,\ f,r,1)$ corresponds to situations in which the robot sidesteps at v_j but not at v_{j+1} and symmetrically for $u_{NY}(j,k,l,\ f,r,1)$. Vertex $u_Y(j,k,l,\ f,r,0)$ has three outgoing arcs for each possible number h of obstacles to be moved outward: one arc corresponds to the h-out extension that moves an obstacle to the sidestep vertex of v_{j+1} (if it exists) and thus it is directed to a u_{YN} vertex; the other two arcs correspond to h-out extensions that do not move obstacles to the sidestep vertex of v_{j+1} and thus they are directed to vertices u_{YY}. The obstacles moved by the two out-extensions do not necessarily coincide, as we shall see briefly in the following. Vertex $u_N(j,k,l,\ f,r,0)$ is connected in a similar way. Vertex $u_{YY}(j,k,l,\ f,r,1)$ has one outgoing arc for each (h_1,h_2)-back extension of $\mathcal{C}(j,k,l,\ f,r,1)$. Vertex $u_{NY}(j,k,l,\ f,r,1)$, instead, has only one outgoing arc corresponding to a back-extension that does not move vertices backward. In a similar way we define the arcs outgoing from $u_N(j,k,l,\ f,r,0)$, $u_{NY}(j,k,l,\ f,r,1)$, and $u_{NN}(j,k,l,\ f,r,1)$.

Let us discuss now how these extensions are computed. Back extensions are computed as described in Section 5, possibly considering the sidestep of v_{j+1} as the destination of a forward move and the origin of a backward move and the sidestep of v_r as the destination of a backard move. The procedure for computing the out-extensions corresponding to arcs going to vertices u_{YN} and u_{NN} is similar to the procedure given in Section 5, but it has to consider also the sidestep of v_{j+1} as a possible destination of outward moves. Instead, a more accurate description of the procedure for computing the out-extensions corresponding to arcs going to vertices u_{YY} and u_{NY} is necessary.

Let v be the sidestep of the branch vertex v_{j+1}. We start by observing that v must be empty when the robot comes to v_{j+1}. Thus, if $v \in O$ the obstacle at v must be moved before the robot comes to v_{j+1}; if v is a hole, instead, it can be the destination of a forward move and the origin of a backward move, but this backward move must be performed before the robot comes to v_{j+1}. In both cases the moves starting and ending at v should be part of the left part of $\mathcal{C}(j,k,l,\ f,r,0)$, while all the left parts of $\mathcal{C}(j,k,l,\ f,r,0)$ do not include any such move.

An approach to this problem could be to add to the left parts of $\mathcal{C}(j, k, l, f, r, 0)$ the obstacle moves involving v. However, it can be easily seen that we cannot simply add moves, since the new moves could possibly violate the properties of LCP. For example, if $L \in \mathcal{C}(j, k, l, f, r, 0)$ contains a forward move to a vertex $w > v_{j+1}$ and no obstacle is moved forward to v, then the resulting plan violates property 3 of LCP. To fix this problem we give out-extensions the power to "replace" some moves of the left parts. Obviously, the new moves introduced need to preserve the properties of the LCP.

We distinguish two cases, depending on whether $v \in O$ or not.

v is a hole. Let $L \in \mathcal{C}(j, k, l, f, r, 0)$ and let A be a LCP such that $L(A, j) = L$. We notice that if L contains at least one forward move ending at a vertex $w > v_{j+1}$ then it must contain also a forward move ending at v, otherwise A would violate property 5 of LCP. Thus, we have to replace some moves of L so that it is possible to include the needed forward move while satisfying the properties of LCP. This replacement involves only moves starting and ending at vertices between v_{j+2} and v_l.

First of all, we observe that a replacement is necessary only if $k > 0$ and $l > j + 1$ or if $f > j + 1$. In all other cases it can be easily seen that there is no pair of moves of L that violates property 5 of LCP. The out-extension works in two steps: it first performs the replacement and then computes the outward moves on the new left part obtained after the replacement (possibly belonging to a different class). We can consider two different replacement policies that add a move ending at v while preserving the LCP properties, depending on whether the number of suspended obstacles remains the same or not. We compute two different h-out extensions corresponding to the two replacement policies: the first out-extension reduces by one the number of suspended obstacles (and thus it modifies the values of k and l but not f); the second replaces the moves so that the number of suspended obstacles does not change (and thus it modifies the value of f but not k and l).

Consider first the case in which we decrease the number of suspended obstacles (notice that in this case we are assuming that $k > 0$). Suppose that L contains $m > 0$ forward moves ending at vertices between v_{j+2} and v_f and let $u_i \to w_i$ be these forward moves, with $1 \le i \le m$ and $w_1 < w_2 < \cdots < w_m = v_f$. The out-extension M replaces the forward move $u_1 \to w_1$ with the forward move $u_1 \to v$ and for each $2 \le i \le m$ the move $u_i \to w_i$ with the forward move $u_i \to w_{i-1}$. Moreover, it adds a new forward move $v_j \to w_m$. If $m = 0$, instead, M simply adds to L the forward move $v_j \to v$.

Consider now the moves of L originating between v_{j+2} and v_l. With a little abuse of notation we denote both outward and backward moves in the same way. We observe that we replace outward and backward moves of L with new moves having the same destinations. Thus, replacing the move $u \to w$ with the move $z \to w$ means that the new move ending at w is of the same type of the old move ending at the same vertex (if the move is backward it is performed while the robot is at the same sidestep vertex). As for the previous case, let $u_i \xrightarrow{z_i} w_i$, with $1 \le i \le m$, and $u_1 < u_2 < \cdots < u_m = v_l$ be the moves of L originating at vertices between v_{j+2} and v_l. M replaces the move $u_1 \to w_1$ with the move $v \to w_1$ and the moves $u_i \to w_i$, $1 \le i \le m - 1$, with the moves $u_i \to w_{i+1}$. Notice that, in the left part obtained after the replacement no move starts from v_l. The canceled move will still be considered as a part of a left part belonging to a class with a greater value of l. This is potentially dangerous for the case in which $l = d$ as there is no class with a greater value of l that can contain the removed move from v_l. To fix this we add d full vertices to the end of the critical path P and consider classes with values of l up to $2d$. In this way even if all vertices of the critical path are branch vertices and each out-extension contained in a LCP causes the removal of a backward move all the possible moves are still considered.

The out-extension which does not change the number of suspended obstacles is constructed in a similar way. However, in this case we have to replace only the moves originating between v_{j+2} and v_f.

We observe that in both the cases the replacement process changes only destinations of forward moves and origins of backward or outward moves. Thus, the replacement can be done by knowing only the values of j, k, l, f, and r and the distributions of the holes in the path between v and v_l. Moreover, all the LCP whose left parts belong to $\mathcal{C}(j, k, l, f, r, 0)$ are replaced in the same way. The cost of the replacement can be computed without knowing the real destinations of the moves replaced.

v is not a hole. The move of the obstacle at v can be performed either in an out-extension (in which case it is sent outward) or in a back-extension (in which case either it is sent backward or it becomes a suspended obstacle).

Let A be a LCP that takes the robot to v and such that $L(A, j)$ belongs to $\mathcal{C}(j, k, l, f, r, 0)$. Obviously, A contains an obstacle move originating at v. We state that if $l > j + 1$ then this move must be part of $L(A, j)$ or of one of its extensions. In fact, if this not the case then v is moved to a vertex $w \geq v_{j+1}$, thus forming a 6-problematic pair with v_l that is moved in L.

The algorithm considers first the possibility of moving the obstacle outward to B_{j+1} or backward to $\mathrm{Ch}(r, j)$. If this is not possible it replaces some moves of L to add a move starting at v while preserving the properties of LCP. In particular, if M is a h-out extension, with $h > 0$, then the obstacle at v is moved outward to B_{j+1}. If M is a 0-out extension outgoing from a vertex y_{YY} then the obstacle at v is moved backward to $\mathrm{Ch}(r, j)$. This move will be performed during the following back-extension. Thus, we need a way to encode the fact that the obstacle at v has been moved or not as part of an out-extension so that the back extension can possibly take in consideration the move starting at v This is achieved by doubling the vertices of the type $u_{YY}(j, k, l, r, 1)$: one $u_{YYY}(j, k, l, r, 1)$ for the case in which the obstacle has been moved (this vertex is connected through h-out-extensions with $h > 0$) and one $u_{YYN}(j, k, l, r, 1)$ for the case in which the obstacle has not been moved (this vertex is connected through h-out-extensions with $h = 0$).

Finally, if M is a 0-out extension outgoing from a vertex y_{NY} then M has to replace some moves of $L(A, j)$ to take care of the obstacle at v. The replacement process is similar to that described in the previous paragraph. The cost of the replacement depends only on the values of j and l and can be computed in constant time.

If $l = j, j + 1$, instead, there is no need for replacing some of the moves of the left part. The obstacle at v will be moved by one of the extensions of L according to the same algorithm of the previous case.

ACKNOWLEDGMENTS

The authors thank Mimmo Parente for his collaboration in the early stages of this research and for many fruitful discussions.

REFERENCES

1. V. Auletta, A. Monti, D. Parente, and G. Persiano (1996), *A linear time algorithm for the feasibility of pebble motion on trees, Algorithmica*, to appear. A preliminary version appeared in Proceedings of Scandinavian Workshop on Algorithm Theory, (SWAT) Lecture Notes in Computer Science, Vol. 1097, pp. 259–270, Springer-Verlag, Berlin/New York.
2. O. Goldreich (1993), "*Shortest Move-Sequence in the Generalized 15-puzzle is NP-hard*," Technical Report 792, Computer Science Department, Technion.
3. D. Kornhauser, G. Miller, and P. Spirakis (1994), *Coordinating pebble motion on graphs, the diameter of permutations groups, and applications, in* "Proceedings of 25th IEEE Symposium on Foundations of Computer Science, (FOCS)," pp. 241–250.
4. C. Papadimitriou, P. Raghavan, M. Sudan, and H. Tamaki (1994), *Motion planning on a graph, in* "Proc. of 35th IEEE Symposium on Foundations of Computer Science, (FOCS)," pp. 511–520.
5. D. Ratner and M. Warmuth (1990), *Finding a shortest solution for the* $(N \times N)$*-extension of the 15 puzzle is NP-hard, J. Symbolic Comput.* **10**, 111–137.
6. J. T. Schwartz, M. Sharir, and J. Hopcroft (1987), "*Planning, Geometry and Complexity*," Ablex, Norwood, NJ.
7. R. M. Wilson (1974), *Graph puzzles, homotopy, and the alternating group, J. Combin. Theory (B)* **16**, 86–96.